

1988 Goddard Conference on Space Applications of Artificial Intelligence

*Proceedings of a conference held at
NASA Goddard Space Flight Center
Greenbelt, Maryland
May 24, 1988*

NASA

NASA Conference Publication 3009

1988 Goddard Conference on Space Applications of Artificial Intelligence

Edited by
James Rash and Peter Hughes
Goddard Space Flight Center
Greenbelt, Maryland

Proceedings of a conference held at
NASA Goddard Space Flight Center
Greenbelt, Maryland
May 24, 1988



National Aeronautics
and Space Administration

Scientific and Technical
Information Division

1988

Foreword

This document, The Proceedings of the 1988 Goddard Conference on Space Applications of Artificial Intelligence, contains a diverse array of papers representing the expanding utilization of Artificial Intelligence (AI) in the space program. We are pleased to present an impressive selection of work, not only from Goddard and other NASA centers, but also from universities and industry engaged in this field. The Third Annual Goddard Conference on Artificial Intelligence provided an opportunity for researchers and practitioners to present their work to others, to compare the effectiveness of various approaches, and to discuss common interests and goals within the AI community.

As we enter the space station era of computing, data systems are becoming more complex and, hence, demanding more advanced and sophisticated methods to solve the associated problems. AI has provided solutions to many problems that were previously difficult or even unsolvable using conventional techniques. Although the true potential of Artificial Intelligence has yet to be realized, the current utility of this rapidly advancing field has proven to be highly beneficial to the space program.

To honor the time and effort that the authors and presenters applied to the conference, we presented two awards on the day of the conference: one for the Best Paper and another for the Best Presentation. We would like to congratulate R. Bowin Loftin (University of Houston-Downtown), Lui Wang and Paul Baffes (NASA/Johnson Space Center), and Grace Hua (Computer Sciences Corp.) who received the Best Paper award for their paper entitled "An Intelligent Training System For Space Shuttle Flight Controllers". We would also like to congratulate Amy Geoffroy and Daniel Britt of Martin Marietta Information and Communications Systems who were awarded the Best Presentation award for their presentation on "Contingency Rescheduling of Spacecraft Operations".

The conference would not have been so successful without the dedicated efforts of many people. First, we would like to thank the members of the judging committees who had the formidable task of choosing the Best Paper and Best Presentation. Second, we thank the conference committee for the endless time and effort they contributed to the conference. Third, we thank the Mission Operations and Data Systems Directorate for sponsoring the conference again this year. Finally, we would like to thank those who truly made the conference possible: the authors whose research and development efforts are presented here and who so energetically conveyed their work to the attendees.

Peter M. Hughes
James L. Rash
Co-Chairmen, 1988 Goddard Conference on Space Applications of Artificial Intelligence

PRECEDING PAGE BLANK NOT FILMED

Acknowledgments

Awards Presentation

Dr. John W. Townsend, Jr., Center Director

Best Presentation Judging Panel

John Dalton, GSFC
Joseph Rothenberg, GSFC
Dr. John Dorband, GSFC
Dr. Andrew Sage, George Mason University
Dr. Donald Perlis, University of Maryland

Best Presentation Finalists Selection Panel

William Macoughtry, GSFC
Dorothy Perkins, GSFC
Larry Hull, GSFC
Joy Bush, Computer Sciences Corp.
David Beyer, Bendix Field Engineering Corp.
(also a supplemental judge for Best Paper)

Conference Committee

James Rash (Co-Chairman), GSFC
Peter Hughes (Co-Chairman), GSFC
Dorothy Perkins, GSFC
Carolyn Dent, GSFC
Daniel Mandl, GSFC
Robert Dominy, GSFC
Troy Ames, GSFC
Michael Bracken, RMS, Inc.

Table of Contents

<i>Mission Operations Support</i>	1
An Intelligent Training System For Space Shuttle Flight Controllers R. Bowen Loftin, Lui Wang, Paul Baffes, Grace Hua	3
Artificial Intelligence Costs, Benefits, Risks For Selected Spacecraft Ground System Automation Scenarios Walter Truszkowski, Barry Silverman, Martha Kahn , Henry Hexmoor	17
A Shared-World Conceptual Model for Integrating Space Station Life Sciences Telescience Operations Vicki Johnson, John Bosley	33
Artificial Intelligence In A Mission Operations And Satellite Test Environment Carl Busse	45
Automated Space Vehicle Control For Rendezvous Proximity Operations Robert Lea	59
Automated Satellite Control In Ada Allan Jaworski, J.T. Thompson	67
<i>Planning and Scheduling</i>	77
Contingency Rescheduling Of Spacecraft Operations Daniel L. Britt, Amy L. Geoffroy, John R. Gohring	79
Knowledge Based Tools For Hubble Space Telescope Planning And Scheduling: Constraints And Strategies Dr. Glenn Miller, Mark Johnston, Shon Vick, Jeff Sponsler, Kelly Lindenmayer	91
The Proposal Entry Processor: Telescience Applications For Hubble Space Telescope Science Operations Robert Jackson, Mark Johnston, Glenn Miller, Kelly Lindenmayer, Patricia Monger, Shon Vick, Robin Lerner, Joel Richon	107
Candidate Functions For Advanced Technology Implementation In The Columbus Mission Planning Environment Audrey Loomis, Albrecht Kellner	125
A Rule-Based Systems Approach To Spacecraft Communications Configuration Optimization James L. Rash, Yen F. Wong, James J. Cieplak	141
Integrated Resource Scheduling In A Distributed Scheduling Environment David Zoch, Gardiner Hall	155

<i>Fault Isolation / Diagnosis</i>	173
MOORE: A Prototype Expert System for Diagnosing Spacecraft Problems Katherine Howlin, Jerry Weissert , Kerry Krantz	175
Achieving Real-Time Performance In FIESTA William Wilkinson, Nadine Happell, Steve Miksell, Robert Quillin, Candace Carlisle	191
Mission Telemetry System Monitor: A Real-Time Knowledge-Based System Samih A. Mouneimne	207
<i>Image Processing and Machine Vision</i>	213
Low Level Image Processing Techniques Using The Pipeline Image Processing Engine In The Flight Telerobotic Servicer Marilyn Nashman, Karen Chaconas	215
Autonomous Image Data Reduction By Analysis And Interpretation Susan Eberlein, Gigi Yates, Niles Ritter	231
An Automated Computerized Vision Technique For Determination Of Three-Dimensional Object Geometry P.T. Chiang, J.C.S. Yang, V. Pavlin	243
An Interactive Testbed For Development Of Expert Tools For Pattern Recognition Stephen W. Wharton	259
Parallel And Distributed Computation For Fault-Tolerant Object Recognition Dr. Henry Wechsler	275
Range Data Description Based on Multiple Characteristics Dr. A.K. Sood, Ezzet Al-Hujazi	295
<i>Data Management</i>	311
The Second Generation Intelligent User Interface For The Crustal Dynamics Data Information System Nicholas Short Jr., Scott Wattawa	313
Spacelab Data Processing Facility Quality Assurance/Data Accounting Expert Systems: Transition From Prototypes To Operational Systems Lisa Basile	329
Automated Cataloging And Characterization Of Space Derived Data William J. Campbell, Larry H. Roelofs, Michael Goldberg	343
A Design For A Ground-Based Data Management System Barbara A. Lambird, David Lavine	355

<i>Modeling and Simulation</i>	371
Automatic Mathematical Modeling For Real Time Simulation System Caroline Wang, Steve Purinton	373
The Space Station Assembly Phase: System Design Trade-offs For The Flight Telerobotic Servicer Dr. Jeffrey H. Smith, Max Gyamfi, Kent Volkmer, Wayne Zimmerman	381
A Simulation Engine - Combining An Expert System With A Simulation Engine James Spiegel, David LaVallee	397
 <i>Development Tools / Methodologies</i>	407
The Advice Taker/Inquirer, A System For High-Level Acquisition Of Expert Knowledge Robert F. Crompt	409
Lisp Object State Saver(LOSS): A Facility Used To Save Partial Schedules Of The Hubble Space Telescope Jeffrey Sponsler	425
Verification and Validation Of Rulebased Systems For Hubble Space Telescope Ground Support Shon Vick , Kelly Lindenmayer	435
The Need For A Comprehensive Expert System Development Methodology Dr. John Baumert, Anna Critchfield, Karen Leavitt	449

Mission Operations Support

**An Intelligent Training System For Space Shuttle Flight
Controllers**

**Artificial Intelligence Costs, Benefits, Risks For Selected
Spacecraft Ground System Automation Scenarios**

**A Shared-World Conceptual Model for Integrating Space
Station Life Sciences Telescience Operations**

**Artificial Intelligence In A Mission Operations And
Satellite Test Environment**

**Automated Space Vehicle Control For Rendezvous
Proximity Operations**

Automated Satellite Control In Ada

**AN INTELLIGENT TRAINING SYSTEM FOR SPACE SHUTTLE
FLIGHT CONTROLLERS**

R. Bowen Loftin
University of Houston-Downtown
One Main Street
Houston, TX 77002

Lui Wang and Paul Baffes
Artificial Intelligence Section, FM72
NASA/Johnson Space Center
Houston, TX 77058

Grace Hua
Computer Sciences Corp.
16511 Space Center Blvd.
Houston, TX 77058

ABSTRACT

An autonomous intelligent training system which integrates expert system technology with training/teaching methodologies is described. The system was designed to train Mission Control Center (MCC) Flight Dynamics Officers (FDOs) to deploy a certain type of satellite from the Space Shuttle. The Payload-assist module Deploys/Intelligent Computer-Aided Training (PD/ICAT) system consists of five components: a user interface, a domain expert, a training session manager, a trainee model, and a training scenario generator. A user interface has been developed which functionally simulates the FDO environment in the MCC. The interface also provides the trainee with information on the characteristics of the current training session and with on-line help (if permitted by the training session manager). The domain expert (DeplEx for Deploy Expert) contains the rules and procedural knowledge needed by a FDO to carry out the satellite deploy. The DeplEx also contains "mal-rules" which permit the identification and diagnosis of common errors made by the trainee. The training session manager (TSM) examines the actions of the trainee and compares them with the actions of DeplEx in order to determine appropriate responses. A unique feature of the TSM is its ability to grant the trainee the freedom to follow any valid path between two stages of the deploy process. A trainee model is developed for each individual using the system. The model includes a history of the trainee's interactions with the training system and provides evaluative data on the trainee's current skill level. A training scenario generator (TSG) designs appropriate training exercises for each trainee based on the trainee model and the training goals. All of the expert system components of PD/ICAT (DeplEx, TSM, and TSG) communicate via a common blackboard. The PD/ICAT system is currently being tested by both experienced and novice FDOs in order to refine the system and determine its efficacy as a training tool. Ultimately, this project will serve as a vehicle for developing a general architecture for intelligent training systems together with a software environment for creating such systems.

PRECEDING PAGE BLANK NOT FILMED

1.0 INTRODUCTION

The Mission Operations Directorate (MOD) at NASA/Johnson Space Center is responsible for the ground control of all Space Shuttle operations. Those operations which involve alterations in the characteristics of the Space Shuttle's orbit are guided by a flight controller, known as a Flight Dynamics Officer (FDO), sitting at a console in the "front room" of the Mission Control Center (MCC). Currently, the training of the FDOs in flight operations is principally accomplished through the study of flight rules, training manuals, and "on-the-job training" (OJT) in integrated simulations. From two to four years is normally required for a trainee FDO to be certified for many of the tasks for which he is responsible during Space Shuttle missions. OJT is highly labor intensive and presupposes the availability of experienced personnel with both the time and ability to train novices. As the number of experienced FDOs has been reduced through retirement, transfer (especially of Air Force personnel), and promotion and as the preparation for and actual control of missions occupies most of the MCC's available schedule, OJT has become increasingly difficult to deliver to novice FDOs. As a supplement to the existing modes of training, the Orbit Design Section (ODS) of the MOD requested that the Artificial Intelligence Section (AIS) of the Mission Support Directorate develop an autonomous intelligent computer-aided training system. After extensive consultation with ODS personnel, a particular task was chosen to serve as a proof of concept: the deployment of a Payload-Assist Module (PAM) satellite from the Space Shuttle. This task is complex, mission-critical and requires skills used by the experienced FDO in performing many of the other operations which are his responsibility.

The training system is designed to aid novice FDOs in acquiring the experience necessary to carry out a PAM deploy in an integrated simulation. It is intended to permit extensive practice with both nominal deploy exercises and others containing typical problems. After successfully completing training exercises which contain the most difficult problems, together with realistic time constraints and distractions, the trainee should be able to successfully complete an integrated simulation of a PAM deploy without aid from an experienced FDO. The philosophy of the PD/ICAT system is to emulate, to the extent possible, the behavior of an experienced FDO devoting his full time and attention to the training of a novice--proposing challenging training scenarios, monitoring and evaluating the actions of the trainee, providing meaningful comments in response to trainee errors, responding to trainee requests for information and hints (if appropriate), and remembering the strengths and weaknesses displayed by the trainee so that appropriate future exercises can be designed.

2.0 BACKGROUND

During the last two decades a number of academic and industrial researchers have explored the application of artificial intelligence concepts to the task of teaching a variety of subjects (*e.g.*, geometry, computer programming languages, medical diagnosis, and electronic troubleshooting). A body of literature is now extant on student models and teaching/tutoring methodologies adapted to intelligent tutoring systems in the academic environment.¹ The earliest published reports which suggested the applications of artificial intelligence concepts to teaching tasks appeared in the early 1970's.^{2,3} Hartley and Sleeman actually proposed an

architecture for an intelligent tutoring system.³ However, it is interesting to note that, in the fifteen years which have passed since the appearance of the Hartley and Sleeman proposal, no agreement has been reached among researchers on a general architecture for intelligent tutoring systems.⁴

Examples of intelligent tutoring systems reported to date are SOPHIE⁵, PROUST⁶ and the LISP Tutor⁷. The first of these systems, SOPHIE, was developed in response to a U.S. Air Force interest in a computer-based training course in electronic troubleshooting. SOPHIE contains three major components: an electronics expert with a general knowledge of electronic circuits, together with detailed knowledge about a particular type of circuit (in SOPHIE this was an IP-28 regulated power supply); a coach which examines student inputs and decides if it is appropriate to stop the student and offer advice; and a troubleshooting expert that uses the electronics expert to determine which possible measurements are most useful in a particular context. Three versions of SOPHIE were produced and used for a time but none was ever viewed as a "finished" product. One of the major lacks of the SOPHIE systems was a user model. It is interesting to note that the development of a natural language interface for SOPHIE represented a large portion of the total task.

PROUST and the LISP Tutor are two well-known intelligent tutoring systems that have left the laboratory and found wider applications. PROUST (and its offspring, Micro-PROUST) serves as a "debugger" for finding nonsyntactical errors in Pascal programs written by student programmers. The developers of PROUST claim that it is capable of finding all of the bugs in at least seventy percent of the "moderately complex" programming assignments that it examines. PROUST contains an expert Pascal programmer that can write "good" programs for the assignments given to students. Bugs are found by matching the expert's program with that of the student; mismatches are identified as "bugs" in the student program. After finding a bug, PROUST provides an English-language description of the bug to the student, enabling the student to correct his error. The system cannot handle student programs that depart radically from the programming "style" of the expert. The LISP Tutor is currently used to teach the introductory LISP course offered at Carnegie-Mellon University. This system is based on the ACT (historically, Adaptive Control of Thought) theory and consists of four elements: a structured editor which serves as an interface to the system for students, an expert LISP programmer that provides an "ideal" solution to a programming problem, a bug catalog that contains errors made by novice programmers, and a tutoring component that provides both immediate feedback and guidance to the student. Evaluations of the LISP Tutor show that it can achieve results similar to those obtained by human tutors. One of its primary features is its enforcement of what its authors regard as a "good" programming style.

3.0 TRAINING VERSUS TUTORING

The PD/ICAT system was developed with a clear understanding that training is not the same as teaching or tutoring.⁸ The NASA training environment differs in many ways from an academic teaching environment. These differences are important in the design of an architecture for an intelligent training system:

- a. Assigned tasks are often mission-critical, placing the responsibility for lives and property in the hands of those who have been trained.
- b. Personnel already have significant academic and practical experience to bring to bear on their assigned task.
- c. Trainees make use of a wide variety of training techniques, ranging from the study of comprehensive training manuals to simulations to actual on-the-job training under the supervision of more experienced personnel.
- d. Many of the tasks offer considerable freedom in the exact manner in which they may be accomplished.

FDO trainees are well aware of the importance of their job and the probable consequences of failure. While students are often motivated by the fear of receiving a low grade, FDO trainees know that human lives, a billion dollar Space Shuttle, and a \$100+ million satellite depend on their skill in performing assigned tasks. This means that trainees are highly motivated, but it also imposes on the trainer the responsibility for the accuracy of the training content (*i.e.*, verification of the domain expertise encoded in the system) and the ability of the trainer to correctly evaluate trainee actions. The PD/ICAT system is intended, not to impart basic knowledge of mathematics and physics, but to aid the trainee in developing skills for which he already has the basic or "theoretical" knowledge. In short, this training system is designed to help a trainee put into practice that which he already intellectually understands. The system must take into account the type of training that both precedes and follows--building on the knowledge gained from training manuals and rule books while preparing the trainee for and complementing the on-the-job training which will follow. Perhaps most critical of all, trainees must be allowed to carry out an assigned task by any valid means. Such flexibility is essential so that trainees are able to retain, and even hone, an independence of thought and develop confidence in their ability to respond to problems, including problems which they have never encountered and which their trainers never anticipated.

4.0 SYSTEM DESIGN

The PD/ICAT system is modular and consists of five basic components:

1. A user interface that permits the trainee to access the same information available to him in the MCC and serves as a means for the trainee to take actions and communicate with the training session manager
2. A domain expert (DeplEx) which can carry out the satellite deployment process using the same information that is available to the trainee and which also contains a list of "mal-rules" (explicitly identified errors that novice trainees commonly make).
3. A training session manager (TSM) which examines the assertions made by the DeplEx (of both correct and incorrect actions in a particular context) and by the trainee. Evaluative assertions are made following each trainee action. In addition, guidance can be provided to the trainee if appropriate for his skill level.

4. A trainee model which contains a history of the individual trainee's interactions with the system together with summary evaluative data.
5. A training scenario generator that designs increasingly-complex training exercises based on the current skill level contained in the trainee's model and on any weaknesses or deficiencies that the trainee has exhibited in previous interactions.

Figure 1 contains a schematic diagram of the PD/ICAT system. Note that provision is made for the user to interact with the system in two distinct ways and that a supervisor may also query the system for evaluative data on each trainee. The blackboard serves as a common "factbase" and communications interface for all five system components. With the exception of the trainee model, each component makes assertions to the blackboard, and the rule-based components look to the blackboard for facts which can "fire" their rules.

4.1 User Interface

The primary factor influencing the interface design was fidelity to the task environment. To avoid negative training, it was deemed essential that the functionality and, to the extent possible, the actual appearance of the training environment duplicate that in which the task is performed. Figure 2 contains a view of the typical display seen by a trainee on a Symbolics 3600 series LISP machine. The upper right corner of the display contains menus that allow the trainee to make requests of other flight controllers, respond to requests from those controllers, call up displays, obtain information about the current or previous step in the deploy process, request help from the training system, and return to a previous step in the process. This menu has as many as three levels depending on the nature of the action taken by the trainee. Some actions are completely menu driven while others require the input of one or more "arguments". All actions taken by the trainee through these menus and the arguments that they may require become assertions to the blackboard. All requests directed to the trainee and all messages sent to the trainee in response to his requests or actions appear in a window in the upper left corner of the screen. These two portions of the screen serve to functionally represent the voice loop interactions that characterize the current FDO task environment. Any displays requested by the trainee appear in the lower portion of the screen, overlapped, if more than one is requested. Clicking the mouse on any exposed portion of a background display will bring it to the foreground. The displays replicate those seen by a FDO "on console" in the MCC. During development nominal data was supplied to these displays (from a dedicated ephemeris-generating program or from "dummy" data sets) so that negative training does not occur. Finally, a "pop-up" window appears approximately in the center of the screen to provide error messages, context information, and help. Experienced FDOs using PD/ICAT have expressed satisfaction with the user interface.

4.2 Deploy Expert

The DeplEx is a "traditional" expert system in that it contains if-then rules which access data describing the deploy environment and is capable of executing the PAM deploy process and arriving at the correct "answers". In addition to "knowing" the right way to conduct the PAM deploy, DeplEx also contains knowledge of the typical errors that are

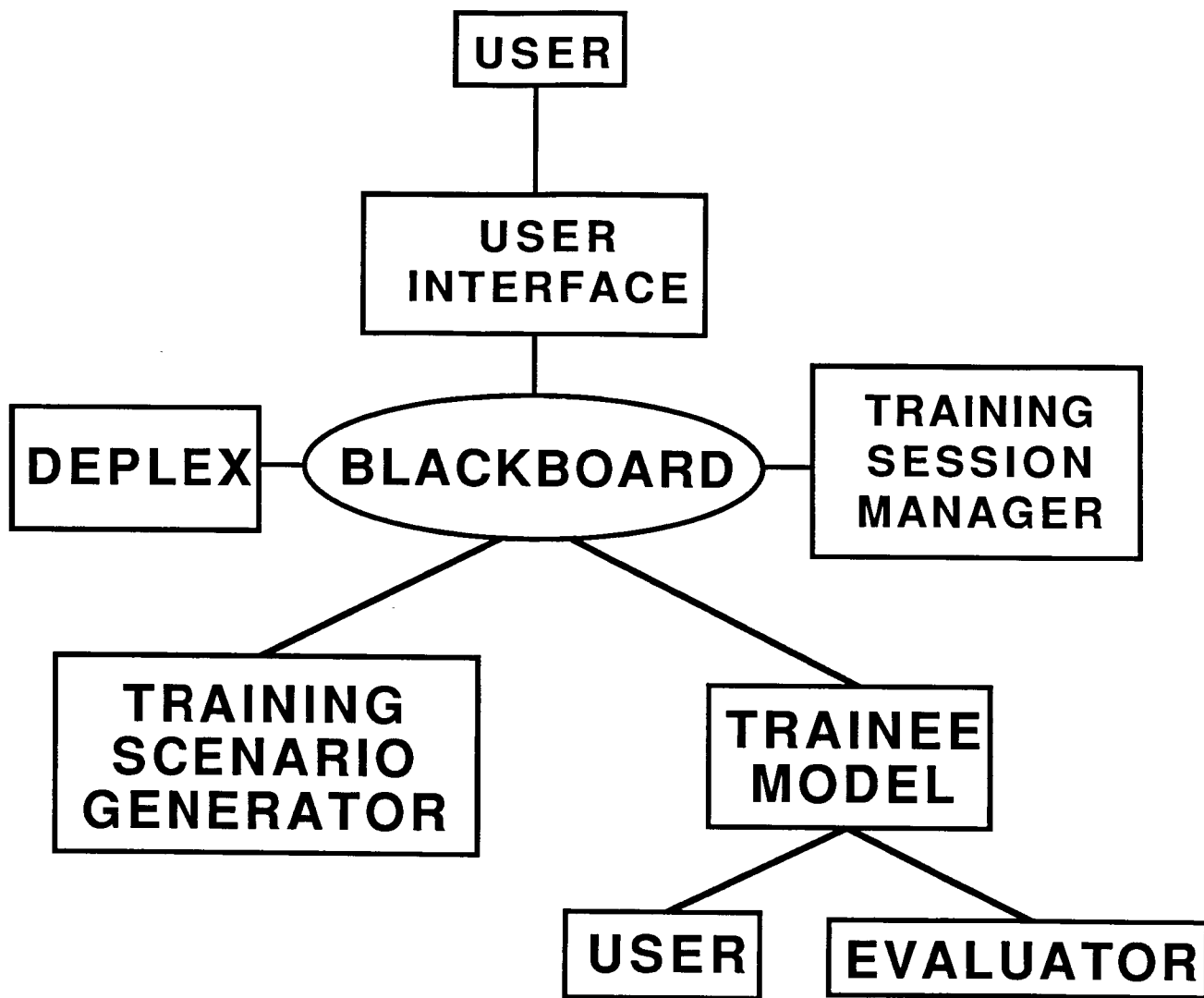


FIGURE 1. PD/ICAT SYSTEM ARCHITECTURE

FIGURE 2. PD/ICAT USER INTERFACE

made by novice FDOs. In this way, PD/ICAT can not only detect an erroneous action taken by a trainee, but also, through these so-called "mal-rules", diagnose the nature of the error. Thus, the system can produce an error message for the trainee specifically designed to inform him about the exact error made and correct the misconception or lack of knowledge which led to the commission of that error. Through interaction with the trainee model, the nature of the message can be adapted to the demonstrated skill level of the trainee (see the following section). Another of the interesting features of the PD/ICAT system is its continual awareness of the environment (the external constraints dictated by the training exercise) and the context of the exercise. This feature provides the basis for "user-directed" behavior on the part of the DeplEx. Rather than DeplEx generating a complete and correct solution to the deployment problem, only those actions which are germane to the current context are asserted. In this way the expert "adapts" to alternate, but correct, paths that the trainee might choose to follow. Figure 3 shows schematically how DeplEx operates. This strategy was adopted because the human experts that perform PAM deploys recognize that many steps in the deploy process may be accomplished by two or more equally valid sequences of actions. To grant freedom of choice to the FDO trainee and to encourage independence on his part, the experts felt that it was essential to build this type of flexibility into the PD/ICAT system.

4.3 Training Session Manager

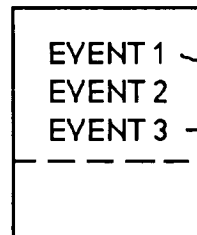
The TSM is dedicated principally to error-handling. Its rules compare the assertions of DeplEx with those of the trainee to detect errors. Subsequently, DeplEx asserts facts that allow the TSM to write appropriate error messages to the trainee through the user interface. In addition, the TSM is sensitive to the skill level of the trainee as represented by the trainee model. As a result, the detail and "tone" of error messages is chosen to match the current trainee. For example, an error made by a first-time user of the training system may require a verbose explanation so that the system can be certain the trainee will have all of the knowledge and concepts needed to proceed. On the other hand, an experienced trainee may have momentarily forgotten a particular procedure or may have "lost his place". In this latter case a terse error message would be adequate to allow the trainee to resume the exercise. The TSM also encodes all trainee actions, both correct and incorrect, and passes them to the trainee model.

4.4 Trainee Model

Successful intelligent tutors incorporate student models to aid in error diagnosis and to guide the student's progress through the tutor's curriculum.⁹ The trainee model in the PD/ICAT system stores assertions made by the TSM as a result of trainee actions. Thus, at its most fundamental level, the trainee model contains, for the current session, a complete record of the correct and incorrect actions taken by the trainee. At the conclusion of each training session, the model updates a training summary which contains information about the trainee's progress such as a skill level designator, number of sessions completed, number of errors made (by error type and session), and the time taken to complete each session. After completing a session, the trainee can obtain a report of that session which contains a comprehensive list of correct and incorrect actions together with an evaluative commentary. A supervisor can access each trainee's model to obtain this same report or to obtain summary data, at a higher

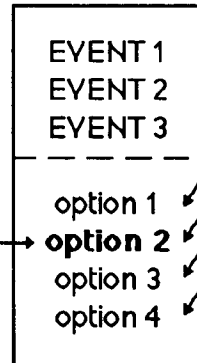
(A) PREVIOUS EVENTS
TRIGGER SECTION
OF DEPLEX CODE

BLACKBOARD

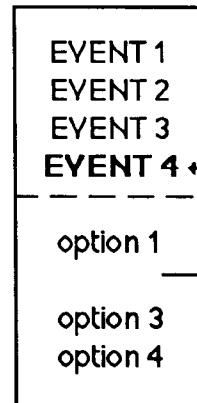


(B) TRAINEE ACTION
MATCHES OPTION
ASSERTED BY
DEPLEX

Trainee Action



(C) MATCHED OPTION
REASSERTED AS
LATEST EVENT



(D) UNUSED OPTIONS
DELETED BEFORE
NEXT STEP

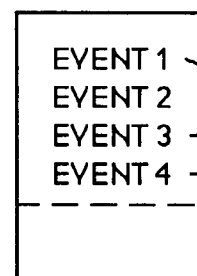


FIGURE 3. DEPLEX OPERATION

level, on the trainee's progress. Finally, the training scenario generator uses the trainee model to produce new training exercises.

4.5 Training Scenario Generator

The training scenario generator relies upon a database of task "problems" to structure unique exercises for a trainee each time he interacts with the system. The initial exercises provided to a new trainee are based on variants of a purely nominal PAM deploy with no time constraints, distractions or "problems". Once the trainee has demonstrated an acceptable level of competence with the nominal deploy, the generator draws upon its database to insert selected problems into the training environment (*e.g.*, a propellant leak which renders the thrusters used for the nominal separation maneuver inoperable and requires the FDO to utilize a more complicated process for computing the maneuver). In addition, time constraints are "tightened" as the trainee gains more experience and distractions, in the form of requests for information from other MCC personnel, are presented at "inconvenient" points during the task. The generator also examines the trainee model for particular types of errors committed by the trainee in previous (and the current) sessions. The trainee is then given the opportunity to demonstrate that he will not make that error again. Ultimately, the trainee is presented with exercises which embody the most difficult problems together with time constraints and distractions comparable to those encountered during integrated simulations or actual missions.

The TSG performs its function by creating an object which represents the parameters needed to define a training scenario. Figure 4 shows the basic structure of the object which is created by the TSG. Note that the TSG may dynamically alter the scenario after the training session has begun in response to rules which it contains. Such dynamic changes to the training scenario are in response to errors made by the trainee which are deemed to require immediate remediation.

5.0 SYSTEM INTEGRATION

The PD/ICAT system is currently operational on a Symbolics 3600 series LISP machine. The user interface and trainee model are written in Common LISP while the rules of DepIEx, TSM, and the training scenario generator are written in ART 3.0. The system will ultimately be delivered to the MOD in a Unix workstation environment. To accomplish this delivery, the ART rules were written to facilitate translation into CLIPS¹⁰ and the LISP-encoded user interface and trainee model will be transferred to the workstation or rewritten in C.

6.0 CONCLUSIONS

The PD/ICAT system has, so far, proven to be a potentially valuable addition to the training tools available for training Flight Dynamics Officers in Space Shuttle ground control. The authors are convinced that the basic structure of PD/ICAT can be extended to form a general architecture for intelligent training systems for training flight controllers and crew members in the performance of complex, mission-critical tasks. It may ultimately be effective in training personnel for a wide variety of tasks in governmental, academic, and industrial settings.

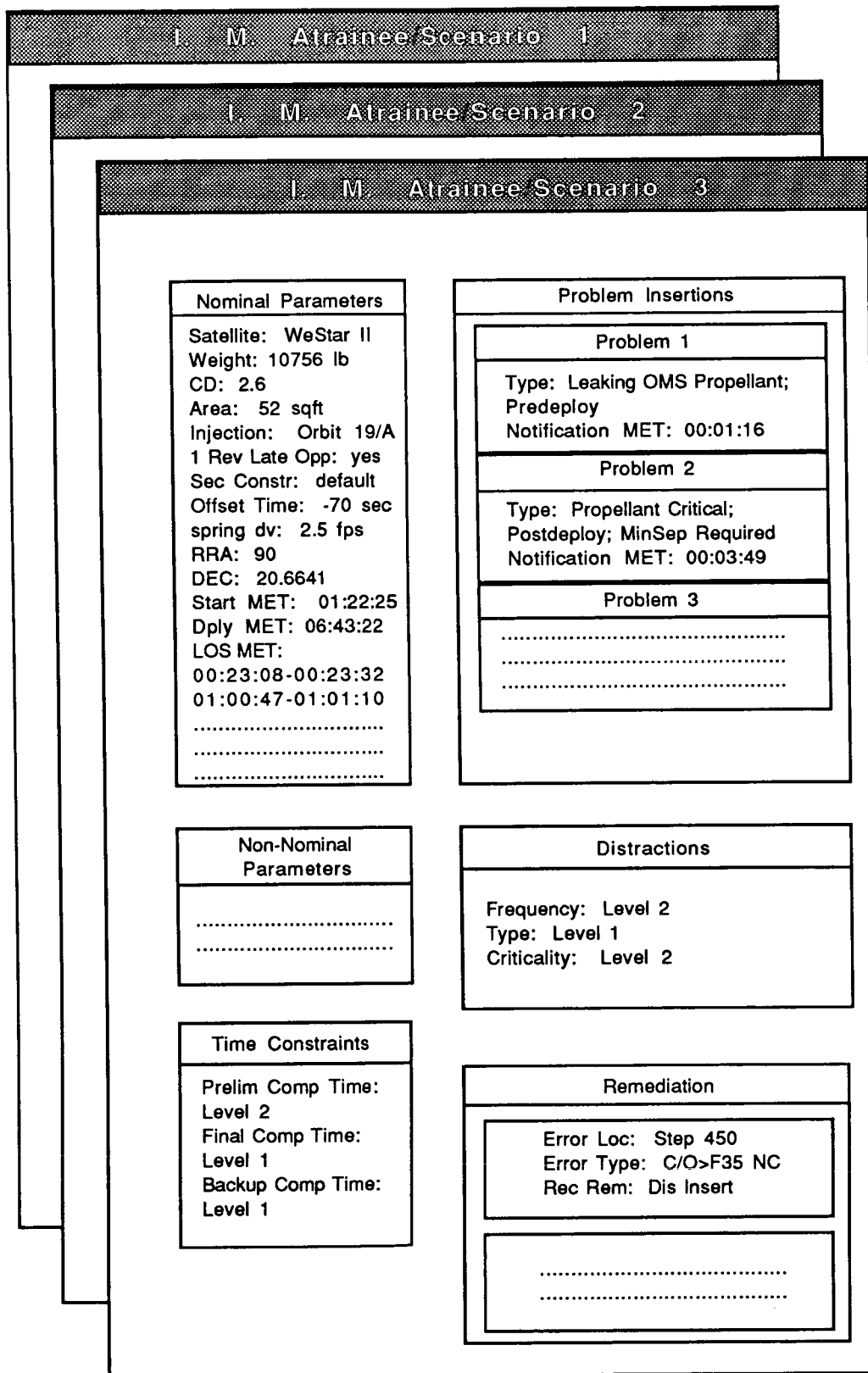


FIGURE 4. SCHEMATIC OF SCENARIO FRAME

ACKNOWLEDGEMENTS

The authors wish to acknowledge the invaluable contributions of expertise from three FDOs: Capt. Wes Jones, USAF; Major Doug Rask, USAF; and Kerry Soileau. Various students assisted with the knowledge engineering and coding of portions of the user interface and TSM: Tom Blinn, Joe Franz, Bebe Ly, Wayne Parrott, and Chou Pham. Finally, the encouragement and guidance of Chiold Epp (Head, ODS) and Bob Savely (Head, AIS) are gratefully acknowledged. Financial support for this endeavor has been provided by the Mission Planning and Analysis Division, NASA/Johnson Space Center and, for RBL, by a NASA/American Society for Engineering Education Summer Faculty Fellowship.

ACRONYMS

AIS	Artificial Intelligence Section
DeplEx	Deploy Expert
FDO(s)	Flight Dynamics Officer(s)
MCC	Mission Control Center
MOD	Mission Operations Directorate
ODS	Orbit Design Section
OJT	on-the-job training
PAM	Payload-Assist Module
PD/ICAT	Payload-Assist Module Deploy/Intelligent Computer-Aided Training
TSM	Training Session Manager
TSG	Training Scenario Generator

REFERENCES

1. See, for example, D. Sleeman and J.S. Brown, eds., Intelligent Tutoring Systems (London: Academic Press, 1982); M. Yazdani, "Intelligent Tutoring Systems Survey," Artificial Intelligence Review 1, 43 (1986); and E. Wenger, *Artificial Intelligence and Tutoring Systems* (Los Altos, CA: Morgan Kaufmann Publishers, 1987).
2. J.R. Carbonell, "AI in CAI: An Artificial Intelligence Approach to CAI," IEEE Transactions on Man-Machine Systems 11 (4), 190 (1970).
3. J.R. Hartley and D.H. Sleeman, "Towards Intelligent Teaching Systems," International Journal of Man-Machine Studies 5, 215 (1973).
4. M. Yazdani, "Intelligent Tutoring Systems Survey," Artificial Intelligence Review 1, 43 (1986).

5. J.S. Brown, R.R. Burton, and J. de Kleer, "Pedagogical, Natural Language and Knowledge Engineering Techniques in SOPHIE I, II, and III," in D. Sleeman and J.S. Brown, eds., Intelligent Tutoring Systems (London: Academic Press, 1982), p. 227.
6. W. L. Johnson and E. Soloway, "PROUST: Knowledge-based Program Debugging," *Proceedings of the Seventh International Software Engineering Conference* (, 1984), pp. 369-380 and W.L. Johnson and E. Soloway, "PROUST," Byte **10** (4), 179 (April, 1985).
7. J.R. Anderson, C.F. Boyle, and B.J. Reiser, "Intelligent Tutoring Systems," Science **228**, 456 (1985) and J.R. Anderson and B.J. Reiser, "The LISP Tutor," Byte **10** (4), 159 (April, 1985).
8. P. Harmon, "Intelligent Job Aids: How AI Will Change Training in the Next Five Years," in G. Kearsley, ed., Artificial Intelligence and Instruction: Applications and Methods (Reading, MA: Addison-Wesley Publishing Co., 1987) p. 165.
9. See, for example, a number of papers on student models in D. Sleeman and J.S. Brown, eds., Intelligent Tutoring Systems (London: Academic Press, 1982).
10. "CLIPS" is an acronym for "C-Language Integrated Production System" and was developed by the Artificial Intelligence Section, Mail Code FM72, NASA/Johnson Space Center, Houston, TX 77058. Its advantages as a delivery vehicle for expert systems are discussed in J. Giarratano, C. Culbert, G. Riley, and R.T. Savely, "A Solution to the Expert System Delivery Problem," submitted for publication in IEEE Expert. For additional information on CLIPS, write to the AI Section at NASA/JSC.

Artificial Intelligence Costs, Benefits, Risks for
Selected Spacecraft Ground System Automation Scenarios

by

Walter F. Truskowski:
GSFC Code 522.1

Barry G. Silverman*
Martha Kahn
Henry Hexmoor
IntelliTek, Inc.

Prepared for
Goddard AI Conference
May 1988

In response to a number of high-level strategy studies in the early 1980s, Expert Systems and Artificial Intelligence (AI/ES) efforts for spacecraft ground systems have proliferated in the past several years primarily as individual small to medium scale applications. It is useful to stop and assess the impact of this technology in view of lessons learned to date and, hopefully, to determine if the overall strategies of some of the earlier studies both are being followed and still seem relevant.

To achieve that end four idealized ground system automation scenarios and their attendant AI architectures are postulated and benefits, risks, and lessons learned are examined and compared. These architectures encompass (1) no AI (baseline), (2) standalone expert systems, (3) Standardized, reusable knowledge base management systems (KBMS), and (4) a futuristic unattended automation scenario.

The resulting Artificial Intelligence lessons learned, Benefits, and Risks for Spacecraft Ground System Automation Scenarios are described.

* - George Washington University

PRECEDING PAGE BLANK NOT FILMED

1.0) Introduction

1.1) **Purpose of Comparison** -- Four idealized ground system automation scenarios are examined in light of lessons learned to date from the application of AI to the aerospace domain. The purpose is to determine which of the feasible alternatives maximize productivity and economic concerns without adversely affecting mission objectives.

1.2) **4 Scenarios Overview** -- The four scenarios evaluated with respect to these issues are now summarized.

(1) Baseline Scenario -- The entire analysis is based on a baseline conceptual design and architecture for a ground system. This scenario is an application of advanced ground system techniques being implemented for the latest spacecraft. All other scenarios are measured as increments relative to the Baseline.

(2) Individual Expert System Scenario -- In this scenario the problem domain is divided into the least coupled subproblems, each worthy of an AI development effort. Separate expert systems are built in a bottom-up fashion for each of the subproblems.

(3) Knowledge Base Management Systems (KBMSs) -- This scenario encompasses the development of a set of standardized, reusable AI/ES components useful for knowledge base management (e.g. automated acquisition, self-organization, built in testing, libraries of inferencing techniques, etc.) that can be used to accelerate the development of, and ease the maintenance and integration of ESs at individual positions of the ground segment.

(4) Unattended Automation (UA) Scenario -- The unattended Automation Scenario is achieved by applying KBMSs (scenario #3) to each individual ES application (scenario #2) and by integrating these into a cooperating whole.

1.4) **Benefit/Risk Assessment Methodology** -- The scenarios are evaluated for lessons learned to date and to determine which maximize productivity and economic concerns without adversely affecting mission objectives.

In addition, the scenarios are evaluated in terms of the recommendations of several high level strategy studies published in the early 1980's.

2.0) Background on Earlier AI/ES Studies for NASA

2.1) **NASA Study Group** -- A report called Machine Intelligence and Robotics: Report of the NASA Study Group was published in 1980. [7] It recommended the large scale application of AI throughout NASA. AI and robotics was recommended for application to mission monitoring, sequencing and control, on-board manipulators, imaging and computer vision, and intelligent sensors. Standardization of software development was emphasized and research into automatic programming was recommended.

2.2) **CODMAC** -- The CODMAC study published in 1982 [5] emphasized (a) real time control of remote sensing systems, (b) uplink capability to acquire data of interest, and to adjust instrument operation, (c) onboard processing for data compression as a user option, (d) rapid distribution of data to users, (e) modular design to ease upgrade in long-lived operation, (f) data systems transportable through all mission phases, and (g) commitment to archiving and distributing data and its supporting information (h) documentation, modularization, and standardization of code and data.

2.3) **RAND/SRI** -- A 1981 SRI report [6] suggests that Space Tracking and Data System (STDS) can benefit from the Artificial Intelligence areas of Expert Systems, Natural-Language Processing, Problem Solving and Planning, and Vision. An Expert System integrated with an AI Planner and a Natural Language Interface was suggested for spacecraft analysis including monitoring and diagnosis. It was felt that such a system would merge well with POCCNET and NEEDS. The study also advocates the development of low-level vision techniques for application in Quality Assurance. Research into the application of vision techniques for registration of ground-control points was also suggested.

2.4) **Silverman and Associates** -- A 1984 report on Space Station era ground system technologies [1] recommended the extensive use of hierarchically integrated and cooperating expert systems to achieve a "Japanese factory" style of autonomous control and productivity enhancement. Unlike the other studies described thus far, this one focused extensively on the actual numerical quantitative costs vs. benefits in terms of staffing, lines of code, budgets, etc.

2.5) **Implications** -- A consensus among these studies emerged in which the opportunity was cited to reduce or at least better support direct human involvement in planning, scheduling, command monitoring, control, diagnosing, and operating positions. In most cases the problem is seen as a large scale software development activity in which standardization and modularization of AI software components appears advisable. Many of the studies also suggested that AI should be coupled with distributed control, increased spacecraft autonomy, and greater experimenter data processing responsibility. These recommendations have not been pursued in an integrated manner as now will be explained in terms of four ground system operations scenarios.

3.0) **Baseline Scenario: Ground System Description (NO AI)** -- The Baseline Scenario Ground System is summarized in the following paragraphs and illustrated in figure 1:

(1) Communications -- Five distinct communication elements are depicted: TDRSS, MODNET, MODLAN, NASCOM and NASCOM switch.

(2) Remote Facility -- The Remote Facility generates command and schedule requests.

(3) Command Management -- Command and schedule requests are converted to S/C constraint-checked command loads by the CMF.

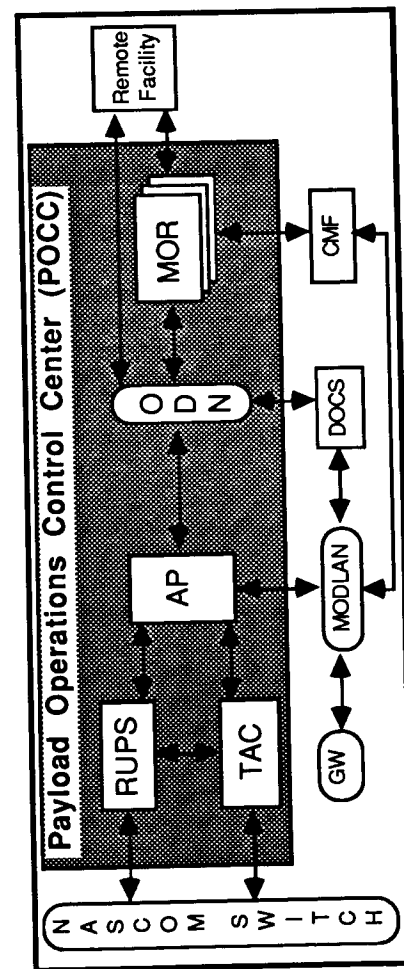
(4) Schedule Management -- Week-long schedule requests for S/C communications are prepared into a draft schedule by the MPT and sent to NCC for final schedule preparation subject to rejection due to conflicts. The MOR prepares individual S/C pass plans.

(5) Science Data Handling -- Ancillary data and science data are sent to the POCC for separation.

(6) Real Time Mission Operations -- The MOR executes the pass plan, monitors ancillary (health and safety) data received by the MSOCC, and instructs MSOCC to transmit command loads.

(7) Off-Line Mission Operations -- Both FDF and SDPF monitor the S/C and on board systems and they produce predictive values for attitude and orbit. FDF also provides sensor calibration information.

(8) The functional components shown as F1 through F9 in Figure 2 [8] are typical of many ground control systems. In general, the key functions performed at the ground facilities can be categorized as: A.) Mission planning, B.) Resource scheduling, C.) Command Management, D.) Command Transmission, E.) Data capture and preprocessing, F.) Data analysis, G.) Monitoring and control, H.) Data archiving, I.) Data distribution, and J.) Simulation and predictions.



ACQO- Acquisition Data	MOR- Mission Op. Room
AP- Applications Processor	MPP- Mission Planning Terminal
ATT- Attitude Data	MSSC- Mission Science Op. Cntrl. Cntr.
CMD- Command Uploads	NCC- Network Control Center
CM- Command Management	ODM- Operation Data Message
CMF- Command Management Facility	ODN- Operation Data Network
DOCS- Data Op. Control System	PSAT- Predictive Satellite Data
FDFF- Flight Dynamics Facility	RUPS- Recorder Utility Proc. System
GCM- Ground Control Messages	SCHED- Schedule
GN- Ground Network	SDPF- Science Data Proc. Facility
GW- Gateway	TAC- Telemetry & Command Processor
MODLAN- Mission Op. Data LAN	TL- Telemetry
MODNET- Mission Op. Data Network	WSC- White Sands Computer

Figure 1 - The NASA End-to-End Data/Information System

Figure 2 and 3 partially illustrates the complexity of the overall ground control function by showing how some of the different data sets are utilized by various functions involved in generic mission ground control. Many data sets participate and interact with one another in order to satisfy the functional and operational requirements for each control function. For example, in Figure 4, the data sets D4 and D5, which are time reference data and sensor orientation data, respectively, are utilized by the function F2, i.e., calibrate and locate observations in order to produce/modify the data set D3, i.e., the sensor calibration data. The figure also illustrates that some functions depend upon other functions and that some ground control objectives can be realized only by a coordination of several system functions.

The Baseline scenario is itself a highly automated design relative to prior GSFC designs due to the use of a number of state-of-the-practice automation techniques for the first time at GSFC, e.g., packetized data switching, extensive electronic networking, and distributed/dedicated institutional facilities and autonomous software for increased Instrument Team self-reliance. It represents a conservative no-fail alternative.

4.0) Individual Expert Systems Scenario: The Baseline Scenario consists of numerous Supervisory Controller Positions (SCPs) defined here as semi-automated workstations operated and controlled by humans. The human plus the computers comprise the SCP.

An SCP ranges in degree of automation from a real time controller sitting at a console display to an offline planner studying computer printouts. SCPs are often arranged and organized in a cooperating though distributed hierarchy of positions and facilities. Supervisory Control Nodes tend to leave the most intellectually arduous cognitive functions up to the human supervisory operator: e.g., the anomaly trouble shooting of the Real Time Controller or the stochastic, plausible reasoning of the Problem Solver & Planner. Even so, with current day Artificial Intelligence (AI) and Expert System (ES) techniques it is becoming increasingly possible to replace the human at the third and highest level of local intelligence.

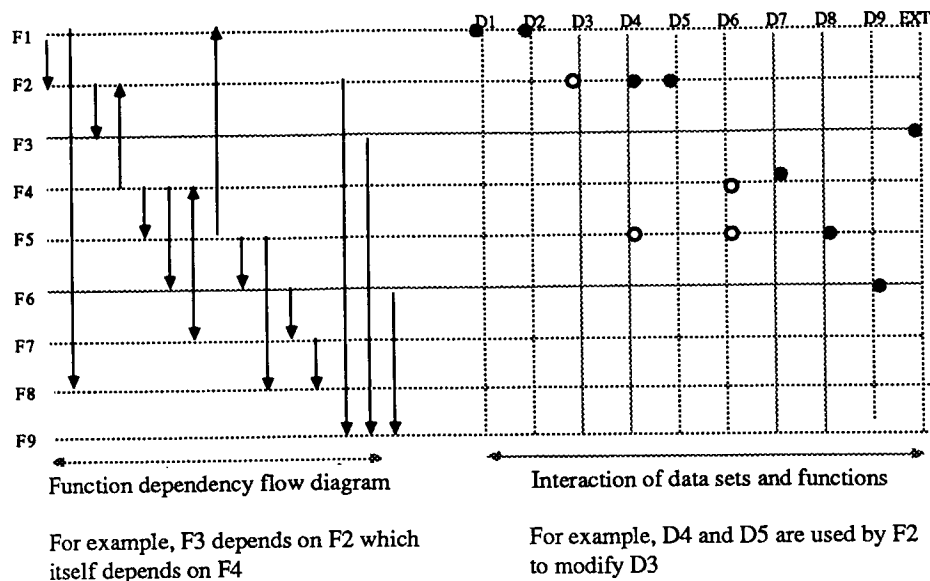
Each of the SPCs at NASA are candidates for individual expert systems. The application of expert systems for replacement of humans monitoring CRT's at NASA was mentioned in several of the studies already cited.* Many such expert systems have been developed over the past several years. Lockheed has developed the Lockheed Satellite Telemetry Analysis in Real Time. (L STAR)

system. Ford Aerospace has developed the Missions Operations Planning Assistant (MOPA). IntelliTek has developed an Expert Project Management System (EPMS). These and a great many other Expert Systems are described in the Proceedings of 1987 Goddard Conference on Space Applications of Artificial Intelligence (AI) and Robotics. Many of these systems are developed to the point where they can advise, rather than replace human monitors, controllers, and planners.

The proliferation of expert systems for spacecraft ground systems in the early eighties has had the benefit that many different approaches have been tried. Similar systems, or systems created for the same function, can now be compared and perhaps new systems proposed which combine the best features of the current systems. Other expert systems may be combined across functional lines, in order to cooperatively solve more complex or broader problems. On the other hand, there has probably been some duplication of effort and failure to share lessons learned by groups working independently. The original study suggestions (section 2 of this paper) are being pursued, however, it seems that it may be too early in the AI technology absorption process to follow the recommendations for standardization and reusability.

While no NASA applications have yet achieved a significant scale, there is an application from which

* RAND/SRI - pg. 16., NASA pg., 36, Silverman & Associates pg. 4-5.



Functions:

- F1- Preprocess Data
- F2- Calibrate and Locate Observations
- F3- Analyze Observations
- F4- Navigate Observatory
- F5- Analyze Observatory Performance
- F6- Plan Mission
- F7- Schedule Resources
- F8- Command Observatory
- F9- Archive Results

Data:

- D1- Raw Observation Data
- D2- Ancilliary Engg. Data
- D3- Sensor Calibration Data
- D4- Time Reference Data
- D5- Sensor Orientation Data
- D6- Spacecraft Attitude Data
- D7- Spacecraft Orbit Observation Data
- D8- Spacecraft Sensor Health and Safety Data
- D9- Science Objectives/Requirements
- Ext- External Data

- Data is modified by Function
- Data is input to Function

Figure 2 Function Interdependency via Data Sets (detailed)

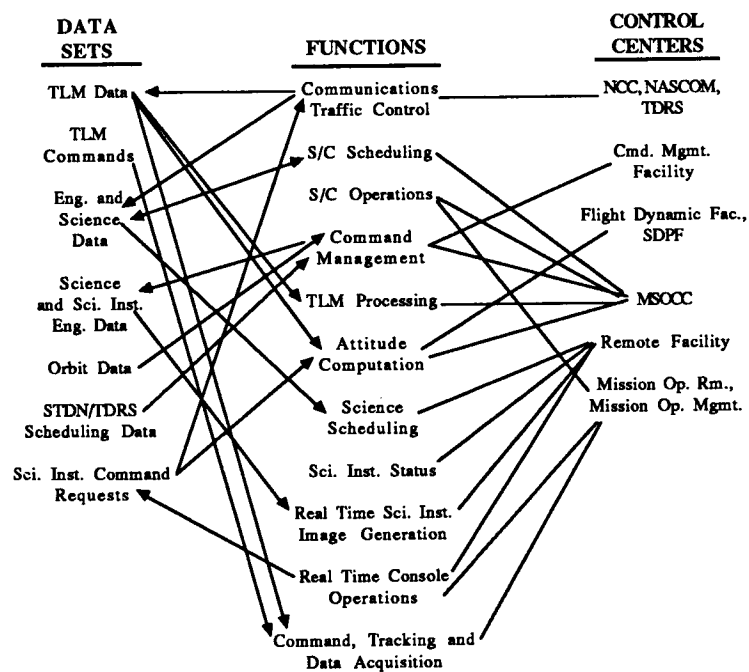


Figure 3 - CONTROL CENTER INTERACTIONS VIA FUNCTIONS AND DATA SETS

"lessons learned" can be inferred. That is the XCON application at Digital Equipment Corp. (DEC). XCON is an Expert System to configure VAX computer systems from a given customer order. The system has knowledge of about 500 components of the VAX and about 4,000 pieces of component information. The system has a conventional architecture with a single Knowledge Base of about 10,000+ rules. Thus the system reflects a fairly large scale ES development effort with the traditional small scale ES architecture. XCON was a successful system in that it helped in dramatically increasing productivity in limited, well-defined classes of customer orders, and the throughput rate was increased significantly. However, during the final test evaluation, XCON failed field test. It was found increasingly difficult to generate and maintain 10,000+ rules. Knowledge bases lack techniques for knowledge elicitation, learning, and self-organization of knowledge which could have avoided problems. Another problem concerns XCON's applicability to a single class of customer order even though all the knowledge about the VAX components could be used for other types of orders as well, for example, VAX clusters. The problem origin may be traced to ineffective usage of knowledge, absence of multiple knowledge bases and lack of new rule generation capabilities. The solution to this problem requires determining requirements for knowledge representation styles and reasoning strategies to enable multiple usage and interpretations of the same knowledge. Another major "nightmare" was found to be a lack of guarantee regarding the consistency of 10,000+ rules. The solution requires Built In Test (BIT) mechanisms such as, but not limited to, case tests, field tests, sensitivity tests, constraint relaxation, bias tests and knowledge base consistency tests. For large scale ES development, a properly integrated architecture and environment is necessary. Scale-up of "single KB type" architecture to suit the needs for a large scale architecture is not sufficient.

5.0) Knowledge Base Management Systems (KBMSs): This scenario allows the individual expert systems (scenario 2) to grow, while avoiding the nightmares experienced during the development of XCON. A key component is the use of standardized reusable parts, as suggested by the high level strategy studies cited in section 2. Useful AI technologies can be seen to share a number of commonalities. That is, much of the core AI technology useful to each individual ES application could be built only once and reused from site to site. A single, shared framework not only would be cost-effective but also would enhance standardization, understandability, and sharing of lessons learned. Such an approach would help current missions as well as others to follow. To this end, an integrated, readily reusable architecture for core AI technology is delineated in terms of three levels of focus: Knowledge Base Management Systems (KBMSs), agent level technology, and distributed problem-solving technology. As shown in Figure 4, these serve as three rings of a generic shell that can be reused from application to application.

At the outermost layer, lies the primary interface that most programmers and users would ultimately have with the system. This ring encompasses the KBMS's two principal interfaces: a) knowledge engineering aids such as application tools, automated test suite, and screen interface utilities; and b) knowledge base and session management that facilitates very large KB organization, storing, swapping, etc. as well as return to and learning from past sessions.

In the next, inner layer of the core AI technology shell, lies a library of techniques for crafting agents tailored to the mission operations industry. These would include the types of representation and logics that are needed but which are difficult to find in off-the-shelf vendor shells. These include many techniques including real time, temporal/situational, constraint propagation, and fusion as well as several others deemed equally important to the GSFC domain such as AI attachable to S/C simulators (model-based); AI attachable to S/C data banks (expert data base systems); and a generic contract formalism language for interfacing to a blackboard.

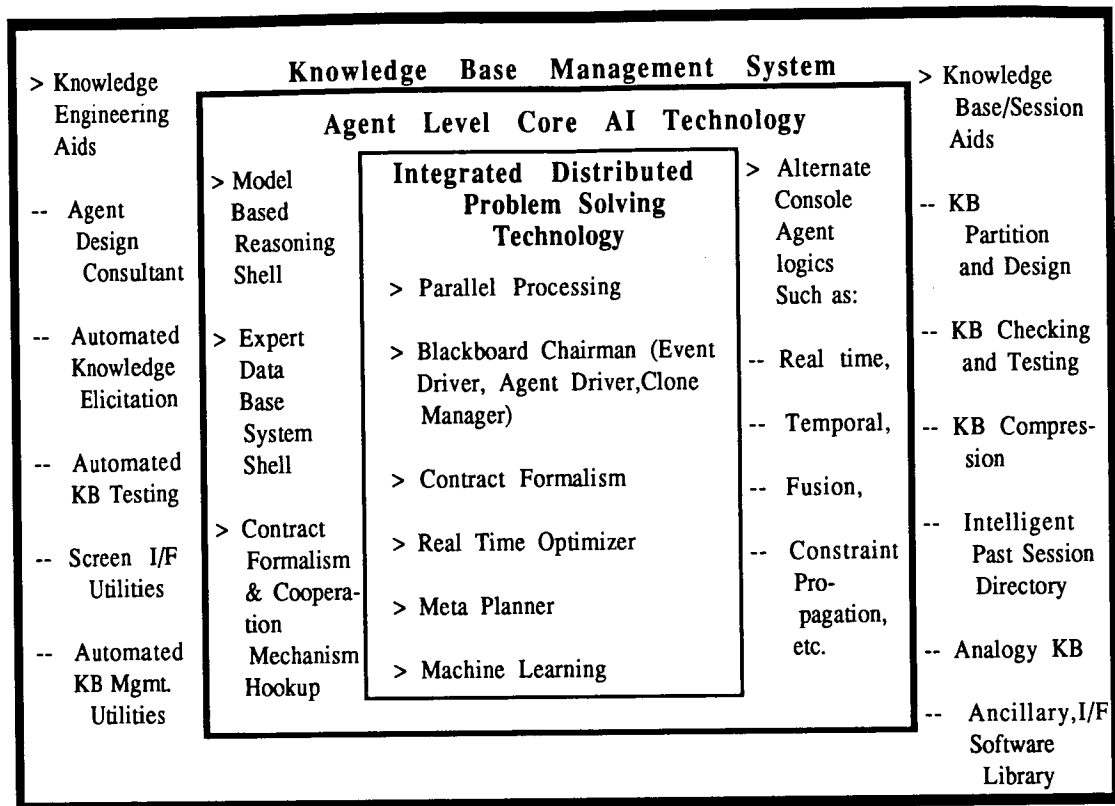


Figure 4 - Overview Of Core AI Environment For GSFC

At the innermost ring, lies the distributed problem-solving technology required to implement a cooperative environment. This includes a multilevel, hierarchical and distributed blackboard capability that is organized into agent-shared knowledge, control level, and meta planning and learning "panels". In addition, there should be a capability for distributed computer resource management to facilitate parallel processing, clone management, and real time response constraints. The individual technologies shown in Figure 4 will now be discussed in more detail.

1. Automated Knowledge Elicitation (AKE): Most current AKE tools have limited ranges of applicability in terms of the types of rules they generate. Their direct relevance to NASA applications needs to be researched and any necessary modifications/extensions created.

2. Automated KB Testing: Testing occurs at many levels in the life of an Expert System ranging from single rule grammar checking to final field implementation testing. A no-fail spacecraft environment demands rigorous, repeatable answers to all test issues before an ES module could be put in service. Automated testing tools are vital if technology insertion is to occur in any reasonable amount of time.

3. Automated KB Management Utilities: KB compression, pointer schemes, memory management, optimal partitioning points, rule relations/hierarchies, etc. are but a few of the topics that need to be examined to flush out a useful KBMS.

4. Alternative Logics and Representations: The design of the first wave of ES shells relied upon backward and/or forward chaining. NASA's applications warrant alternative logics/representations. These include Nonmonotonic Logics, Situational/Temporal Logic, and

Constraint Propagation Techniques.

5. Rigorous Uncertainty Analysis and Fusion Methods: In applications with distributed, noisy sensors, nonmonotonic logics, and multiple human (or AI) agents, fusion of contradictory results must occur. Fusion techniques permit the machine to speculate from a rigorous probabilistic perspective; to update prior judgments; and to merge competing hypotheses while maintaining uncertainty ascribed to each.

6. Expert Data Base Systems: An Expert Data Base System (EDBS) is defined here to be an ES that can formulate DB operations on its own. EDBS's hold two potential roles: 1) To be used on isolated data bases as stand-alone intelligent user interfaces, or 2) to be used as part of a Distributed Expert System which interfaces with and supports database queries/updates.

7. Model-Based Reasoning: Recently model-based reasoning has been extended to allow the ES to model itself to test theories it has postulated. In at least one case, an object-oriented simulation language has also been developed. A distributed agent often needs to test hypotheses on a model of the spacecraft before finalizing conclusions or commands. The direct electronic hookup of a DES to spacecraft models is thus potentially interesting.

8. Clone Management for Parallel Operation: ES agents typically work either in Agent Driven Mode (the agent decides what actions it wishes to pursue next) or Event Driven Mode (the agent is forced to take action by input from the external world). To increase the ability of a given agent to respond to multiple stimuli under either mode it is desirable to create "clones" or duplicate images of the agent and to run each clone in parallel.

9. Parallel Processing Techniques: A topic vital to ES speed up, to clone management, and to contingency analysis is the ability to "parallelize" both symbolic and numerical processing tasks. An important focus for core technology exploration is to identify alternative strategies for algorithm and KB partitioning so as to improve ES performance as number of processors increases.

10. "Real-Time" Design Optimizer: Guaranteed response time is a prerequisite of expert systems in a real time environment. More to the point, discussion on this topic inevitably boils down to "speeding up" expert system technology. The wealth of speed up ideas and techniques available as well as the research and development on these techniques, combined with the constraints of actual ground systems suggest that a single universally applicable speed up design might be unrealistic.

11. Machine Learning and Self-Correction: Machine learning techniques include Learning by Memorization, Learning by Instruction, Learning from Observation, and Learning by Analogy. In each of these types of learning the KBMS plays a major role in facilitating the incorporation, organization, and integration of what has been learned into the existing KBs.

12. Blackboard Technique: All of the core technology described up to this point, is considered to be part of the blackboard. The blackboard can be summarized in terms of Console Agents, Blackboard Panels, the KBMS and the Blackboard Chair. In the blackboard model, each agent is an intelligent, self-organizing Expert System tailored by a specific project (e.g., ST or SS) to perform a narrow set of human console-oriented tasks. The Blackboard chair guides the team of agents toward a shared objective or goal set (also elicited by the KBMS) and orchestrates the pooling of agent insights.

5.2) STCTM: A TESTBED FOR KBMS STUDIES

To explore key features of a large KBMS and to exemplify its incorporation into the scenario, we have developed The Space Telescope Command Telemetry Matcher (STCTM). This is a blackboard

system designed to be a proof of concept prototype and a testbed for exploring large KBMS design issues we discussed in the preceding sections.

The Space Telescope is highly automated, and a single command from a scientist is translated into hundreds of commands before uplink. In addition, the ST internally generates sequences of hundreds of more commands per ground generated command. Similarly, thousands of telemetries are downlinked that collectively indicate execution status of command loads as well as side effects and other onboard activities. STCTM's task is diagnosis and requires assessing as quickly as possible whether the pattern suggested by the flow of 1000s of Command and Telemetry parameters unfolding in real time indicates the scientists desires are being observed.

STCTM scans commands and telemetries and filters them to transform the low level CL/TM to an abstract level. This is accomplished by setting up macro maneuver templates. Each template consists of a set of low level command types. Templates are also referred to as command packets.

STCTM applies two levels of reasoning mechanisms for verification. The initial level is a comparison of expected telemetry with their corresponding actual telemetries. We will refer to this as *cheap test* reasoning. In cases of discrepancy, spacecraft mode is assessed based on the command context and a ST model is consulted for a model expected telemetry. Combinations of command packets, time of command packet uplink, and environmental factors constitute various contexts. This type of reasoning is known as *model based reasoning*.

Currently, STCTM uses simulated commands and it only performs the cheap test. After starting the system, STCTM retrieves the appropriate data and knowledge bases. As part of Blackboard System Generator (BSGTM), user is given the system trace of the blackboard including cycles and specialist activations in a window. The user also observes the current specialist in action in a BSG window for current specialist. In addition, STCTM provides output report incrementally on the blackboard output window.

BSG is a skeletal blackboard development environment and offers the capabilities of commonly available blackboard systems. Additionally, it includes a work breakdown structure to help systematize the design process as well as to allow for goal driven reasoning. It offers reason maintenance techniques to help maintain multiple hypotheses and facilitate parallel reasoning. With STCTM we will attempt to address a number of research issues under the rubric of KBMS including Fusion Methods, Model-Based Reasoning, and Machine Learning as well as rudimentary blackboard enhancements, namely multiple views and hierarchical organization of task domain via work breakdown structure to help control and coordinate problem solving.

We are experimenting with reason maintenance for parallel reasoning. Corrective actions are often time sensitive and a parallel search for resolving the differences is believed to be required. Furthermore, temporal techniques such as time map management will be incorporated to maintain and reason about temporal constraints on commands. This will especially be used for planning corrective actions.

6.0) Unattended Automation Scenario: The Unattended Automation Scenario utilizes robots, Japanese-style industrial process automation, computer aided manufacturing (CAM), automatic control theory with distributed machine intelligence, and enhanced onboard (upstairs) as well as Instrument Team-based (distributed) command and control. Utilizing existing technological approaches, the scenario achieves a classical automated factory situation in which the ground system operates in a manned day shift but with caretaker off-shifts.

The result is a 78% reduction in annual operating staff and costs relative to the Baseline (from 76

positions down to 17 positions) with a simultaneous and significant improvement in achievement of the CODMAC position.[1] Once the KBMS scenario is achieved, this Unattended Autonomy scenario becomes feasible.

Every new era such as the "automated factory" one comes only after substantial development costs and risks. The risks of adapting the automated factory approach to MODSD are minimized via a three step plan whereby (1) the altered version of the Baseline Scenario is constructed; the alteration concerns enhanced upstairs and experimenter capabilities; (2) in parallel, an Automated Systems Testbed is constructed in which one of each of the new technologies is developed, tested, evaluated, and refined; (3) the automated technologies are gradually phased into the altered baseline and the excess staff is gradually phased out during the interval from two years after launch through four years after launch.

6.1) Principles of AI for the Integrated Hierarchy: Unattended Automation in the areas of facility operation is analogous to the automated factory application which has proven so successful in other industries but which has failed as yet to penetrate spacecraft operations except in selected studies and reports, e.g., see [1,2,5,8]. Such a view is fostered in this section.

Spacecraft with their ground systems may be viewed as a manufacturing plant or factory that receives customer orders (command and schedule requests) and transfers, processes, and transforms these into finished products (science and ancillary packets) that are then delivered back to the principal investigators. The factory product in this analogy is science data.

The ES and KBMS technology described in the prior two sections can be seen as contributing to the automated factory design in a bottom-up fashion. The individual expert systems are designed to replace each individual human SPC position, while the KBMS is designed to overcome expected bottlenecks of large scale, multiple ES applications. In this section we present an architecture which would tie multiple SPC-KBMSs into a single integrated, cooperating system based on the automated hierarchical factory concept.

Unlike the baseline scenario, the proposed scenario is devoted to the top-to-bottom control hierarchy needed to move plans, schedules, authorizations and control information down and status information up. The proposed hierarchy is divided into seven levels as described.

- 1) Management Planning (Long Range): Overall decision maker for long range goals, internal investment programs, and users to be serviced.
- 2) Management Planning (Short Range): Implementer of upper management decisions, day-to-day overseeing of investment project progress, coordination of customer schedules and requests for transfer to next lower level.
- 3) Production (Schedule) Controller: Point of interface between management and the various shops for determining overall optimized intershop coordination and schedule.
- 4) Shop Controller: Overall job allocator of workstations, inter-workstation coordinator, and monitor of workstation emergencies/contingencies.
5. Supervisory Workstation (Cells): Determines and sets the local taskings of all operators and equipment under its purview, issues control programs and real-time control instructions to the lower level, recognizes, diagnoses, and responds to emergencies.
- 6) Operator Workstation: Direct interface with the controller system including issuing real-time

instructions; taskings for reacting directly to emergencies either internal to the controller system or externally that may disrupt it; and sending status updates to the supervisor.

7) **Controller System:** This is the equipment and/or software that performs tasks in slavish fashion upon being tasked by the supervisor or operator. Several existing and nonexistent features should be implemented throughout an automated version of a multilevel control model. In particular, nine recommended features are: 1) Closed Loop System, 2) Generic Extendible, Reusable Components, 3) Hierarchical Control, 4) Local Intelligence, 5) State Machines, 6) Control Cycle, 7) Planning Horizon, 8) Hierarchical Scheduling, and 9) Communication by Common Memory.

6.2) Facility Advisor: A Proof of Concept Prototype: An early prototype of the automated factory for ground systems called Facility Advisor was described in 1986. [2] Facility Advisor is a multi-position KBMS, unlike STCTM. The intent is to show a generic expert system approach (1) to certain classes of human supervisory positions commonly encountered in many spacecraft ground facilities (e.g., facility schedulers, workstation operators, and facility hardware/software fault detection positions), and (2) to create a cooperating set of expert systems designed to operate in a loosely coupled hierarchy (i.e., a DES) that can serve as a FACILITY ADVISOR which is a set of ES kernels that potentially can be tailored to any facility. The prototype modeled three positions cooperating within the ground system "factory": (1) the scheduler who decides when equipment and other resources may be allocated to support each user (2) the operator who utilizes the resources to perform a user requested service and who detects and corrects quality problems of the end product (e.g., message code errors, data set noise, etc.) obtaining inputs from the equipment monitor to assist in problem isolation tasks, and (3) an equipment monitor who detects and isolates equipment/resource problems and either corrects them in time for a given service to be completed or suggests an alternative equipment pathway for the scheduler's consideration. In addition, each of these three SCPs individually must interact and cooperate with their counterparts at other control centers/facilities to solve problems and to perform their jobs. The prototype can be described as follows:

The Virtual Machine Hardware -- The virtual machine includes seven parallel boards plus a host. These "boards" are 4 Xerox Lisp machines, one VAX 11/780, and an IBM PC. One of the Lisp machines also contains a PC emulation board. The "boards" are physically connected via an EtherNet except the VAX which is separately connected to the host.

The Virtual Machine Operating System (COP) -- The virtual machines straddle three distinct types of operating systems (MESA, DOS, VMS) with the aid of the COP capability.

FACILITY ADVISOR: Offline Manager -- The Offline Manager position has been prototyped in LISP on a separate machine utilizing an in-house blackboard technology (developed for the ARIEL and EPMS shells). In brief, the Manager places a planning problem or goal on the blackboard, collects and evaluates Specialist Activation Requests (SARs) from the various position specialists who have offered to solve part of the problem, and issues Execution Orders for the Specialists (EOSs) it feels can make the best contributions at the present time. These specialists, in the prototype are on other "boards" of the virtual machine.

Schedule Master -- Objects are created for each piece of hardware, each user service request, service-pathways, and for each of several types of constraints priority, window, service type, etc. The final selection is sent to the Real Time FACILITY ADVISOR for execution (at present the Real Time FACILITY ADVISOR only executes one schedule alternative).

Repairman -- The investigators have collected over 1000 rules used in GSFC Repairman positions, however, only a very simple prototype has yet been implemented. This Repairman uses seven LOOPS rules and 21 LISP functions to operate and monitor one piece of equipment. When it detects

a failure it uses a second set of rules to isolate and correct (or abort) the problem and it reports the results to the Offline Manager.

Operator -- Here too, the investigators have collected over one thousand rules used by GSFC Operators yet only about 15 or 20 rules have been implemented. The Operator is not presently integrated into the FACILITY ADVISOR, however, it demonstrates the detection, isolation, and correction of user request message errors as well as one of the explanation and accounting features.

The Real Time FACILITY ADVISOR -- A situational calculus capable of supporting the DES real time elements has been tested on the VAX with the aid of a DES generator called Hierarchical Control System Environment (HCSE). HCSE provides a language in which to create fast, deterministic production-oriented specialists that communicate with each other via a blackboard mechanism.

This prototype demonstrated: (a) the feasibility of offline planning elements being constructed in different shells, languages, and machines, (b) the role of the Hierarchical Control System Environment (HCSE) for testbedding of real time elements/modules, and (c) the cloning/virtual entity framework.

8.0) Concluding Remarks: The four scenarios discussed in this paper can be viewed as four consecutive stages in the maturation of AI/ES technology in the Spacecraft Operations Industry. The history of Data Base Management Systems (DBMSs) reflects a parallel four stage development.

In the 1950's there was no separation of data and control in DBMSs. This early stage corresponds to the pre-ES Baseline described in this paper. In the 1960's every organizational group managed its own data files, corresponding to the Individual Expert Systems Scenario discussed in Section 4. The 1970's saw a proliferation of DBMSs that included a full spectrum of the functionality which different applications required. The KBMS scenario discussed in Section 5 can be viewed as a similar stage in the development of AI/Es technology. Finally, the 1980's has seen the development of integrated, distributed DBMSs in which separate DBs are able to communicate. This stage is analogous to the Unattended Automation Scenario discussed in Section 6.

The goal of the high-level strategy studies (Section 2) and of AI researchers is to reach the highest stage, Unattended Autonomy. However, the current approach is Individual Expert Systems, and it is almost guaranteed to pose efficiency improvement opportunities. The original AI/ES strategy studies in hindsight thus appear to have been ignored from the perspective of integrated, distributed, reusable approaches.

On the other hand, if one views the current genre of AI/ES applications as a necessary stepping stone, the original strategy studies can be said to have been too ambitious. We must learn to walk before we can run, and many of the applications-to-date have been invaluable from this perspective.

As individual applications grow larger they will undoubtedly begin to encounter the scale-up problems as described above for XCON: (1) knowledge bases too large to verify and that no one person understands any longer, (2) inflexibility in the presence of slightly altered domain conditions, (3) performance degradation concerns associated with slowness, interfaces, etc.

The original strategy studies may have painted a picture that was too ambitious to reach in a single step. Nevertheless, the picture they painted would seem to be one that must be striven for in a domain as complex, interconnected, and large-scale as GSFC ground systems operations. For these reasons we feel that research and development of advanced Knowledge Base Management System (KBMS) techniques, standards, and reusable modules appears vital for Goddard's future eras. Some of the questions that need investigating, among others, include: How can heterogeneous KBs best be integrated? What are the desirable technologies for acquiring, organizing, compressing, and

storing very large, distributed knowledge bases? Which fusion techniques will prove most effective for combining conflicting information across supervisory controller positions? Can inconsistencies in different KBs (or even in a single large KB) be effectively and reliably detected via automated tools such as built in test? How can the reliability of heuristics, opportunistic, and/or parallel beliefs be assessed and improved? Can generic symbolic, explanation-based learning (rather than unexplainable neural nets) be evolved for rule discovery and knowledge generation purposes?

Such an agenda of AI research issues will not be solved any time soon. However, the need to confront the more difficult AI topics exists and will only grow more prevalent at GSFC as the existing applications begin to mature.

REFERENCES

- [1] Silverman, B.G., Moustakis, v.s., Robless, R.L., Machine Intelligence and Robotics in the Space Station Era, IAI, G.W. University, Washington, D.C., October 1984
- [2] The Facility Advisor: A distributed Expert System testbed: Functional Requirements Document and Plan, Prepared under NAS5-28604, Technical Monitor and GSFC Code 522.1 (W.F. Truskowski), June, 1986.
- [3] Silverman, B.G., Hexmoor, H.H., Rastogi, R., Core Artificial Intelligence (A.I.) Technology and Space Telescope Autonomy, Prepared under NAS5-30037, Technical Monitor and GSFC code 522.1 (W.F. Truskowski) August 1987
- [4] Truskowski, W. Silverman, B., Hexmoor, H.H., Rastogi, R., A Design Methodology For Knowledge Base Management Systems, July 10, 1987.
- [5] Committee on Data Management and Computation. Space Science Board. Data Management and Computation. Vol.1 Issues and Recommendations. National Academy Press. Washington, DC 1982.
- [6] Brown, David R., Applications of Artificial Intelligence in Space Tracking and Data Systems, Task 1 Report, Contract NAS5-26358, SRI Project 2203, SRI International Menlo Park, CA
- [7] Sagan, Carl Machine Intelligence and Robotics: Report of the NASA Study Group. NASA, 1980
- [8] Data Base System Architecture Study, By Computer Technology Associates, Inc., for NASA, GSFC NAS5-26893, February 1983.
- [9] Silverman, Barry G. FACILITY ADVISOR: A Distributed Expert System Testbed for Spacecraft Ground Facilities, Institute for Artificial Intelligence, George Washington University

A Shared-World Conceptual Model for Integrating Space Station Life Sciences Telescience Operations

Vicki Johnson
Research Institute for Advanced Computer Science
John Bosley
Bionetics Corporation

NASA Ames Research Center
Moffett Field, CA 94035

Abstract:

Mental models of the Space Station and its ancillary facilities will be employed by users of the Space Station as they draw upon past experiences, perform tasks and collectively plan for future activities. The operational environment of the Space Station will incorporate telescience, a new set of operational modes. To investigate properties of the operational environment, distributed users and the mental models they employ to manipulate resources while conducting telescience, this paper proposes an integrating **shared-world** conceptual model of Space Station telescience operations. The model comprises distributed users and resources (*active elements*); *agents* who mediate interactions among these elements on the basis of intelligent processing of shared information; and *telescience protocols* which structure the interactions of agents as they jointly accomplish operational *tasks*. Intelligent agents utilize views of the shared world as they engage in cooperative, responsive interactions on behalf of users and resources distributed in space and time. An agent's behavior may range from standardized to idiosyncratic, from naive to intelligent, but the requirement for a common world view, communicated through standardized telescience protocols, remains essentially invariant. This model permits partitioning of knowledge, processing and control between active elements, agents and the infrastructure supporting telescience. Examples from the life sciences are used to instantiate and refine the model's principles. Implications for transaction management and autonomy are discussed. Experiments employing the model are described which the authors intend to conduct using the Space Station Life Sciences Telescience Testbed currently under development at Ames Research Center.

Introduction

The Space Station will be a multifunction, long-term facility for living and doing science in space. Users of Space Station resources (both ground-based and space-borne) will often be distributed and disparate. Many different technologies will be utilized to connect distributed, cooperating users to their remote resources; these new modes of operation, and their enabling technologies, have been termed "telescience" [1]. The activities to be performed aboard the Space Station will vary widely, from sophisticated science experiments guided by strict protocols to routine maintenance operations. On the ground, coordinated planning, rerouting and analysis of data must be accomplished across distributed facilities. The many physical, temporal, and cognitive dimensions inherent in Space Station tasks will make sustained operation impossible unless there is constructed and referenced an integrating conceptual model of the goals and capabilities of the Space Station *world* (the dynamic, operational environment) and shared views of its *task domains*. Presented here is a model of users and resources linked together into a communicating, cooperating whole which has a goal of performing tasks consistent with the conventions and objectives of their shared world.

The model comprises distributed users and resources (*active elements*); *agents* who mediate interactions among these elements on the basis of intelligent processing of shared information; and *telescience protocols* which structure the interactions of agents as they jointly accomplish operational *tasks*. This paper presents an analysis of some of the properties the agents and telescience protocols must possess to accomplish prototypical life sciences operational tasks, such as user/user collaboration and user/resource monitoring. Although the concept of shared-world views as a basis for telescience is perfectly general, the life sciences discipline with which the authors work imposes several unique requirements on distributed shared-world views, and these characteristics are used to illustrate and refine the model in this paper. The authors assert that issues of transaction management and autonomy are of particular importance to the life sciences, and support this assertion by examining the roles of autonomous versus teleoperated agents in the life sciences. Finally, experiments based on the model are discussed which the authors intend to conduct using the Space Station Life Sciences Telescience Testbed currently under development at Ames Research Center.

Motivations for the Model

The end-to-end architectures for the Space Station Information System (SSIS), Space Applications and Information System (SAIS) and Life Science Information System (LSIS) are service based designs: dynamic server-client sessions are established to provide services such as resource allocation [2,3]. For example, the SSIS offers communication, management and application services, such as file transfer and transaction management. However, infrastructure for preserving the character of the information system (e.g., its history, purposes, shortcomings and usage conventions) for use across sessions is not generally an explicit part of the architectural designs, nor are mechanisms for codifying such attributes as session parameters. Similarly, discussions of telescience as a mode of operation have in the past been

generally limited to specific applications and interactions (e.g., telerobotics), or constituent information system elements (e.g., needed teleoperation functionality).

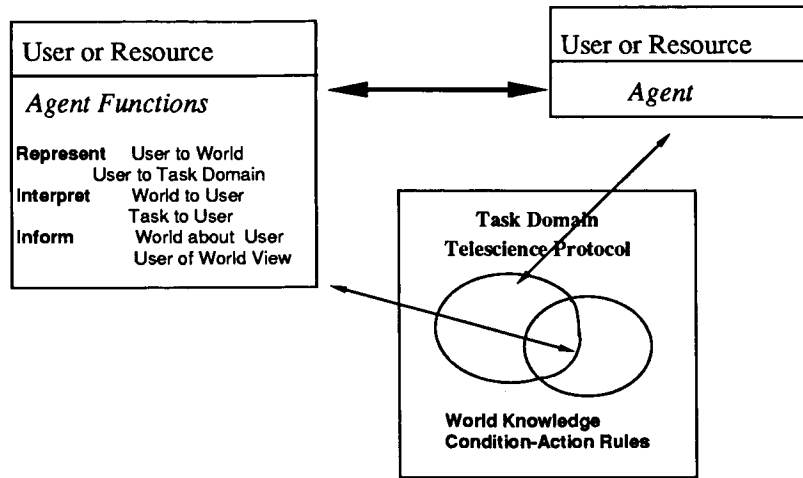
It is the authors' contention that the complexity of Space Station operations for life sciences requires considerable contextual (world) information be presented to distributed users and intelligent, automated resources (such as robots) to enable efficiently structured and coordinated interactions. Agents must have not only a functional understanding of Space Station technologies, but also a *holistic* view of the activities in which the technologies participate [4]; to achieve this, the model makes explicit use of shared-world views. Users and resources engaged in telescience use views of shared system capabilities, configurations and plans to condition the conduct and predict the outcome of their tasks. These views must be presented in a representational form that matches the "intelligence", or information-processing capacity of the active element. Thus the responsibilities of an agent go well beyond the SSIS/SAIS/LSIS server/client role: in their task domain, agents mediate the interface between the system's periphery -- sensors and effectors, users and resources -- to assure that transactions with the real world are consistent with the goals and intentions built into an "ideal" world view. The use of views, which corresponds to the construction of mental models to integrate knowledge and achieve system goals, is an important source of inductive change in long-term knowledge [5,6].

Telescience protocols incorporate shared-world views into the performance of operational tasks and enforce orderly, standardized interactions between agents. Telescience protocols structure and supervise agent interactions, access and update the base of world knowledge (and possibly its rules of operation) and obtain infrastructure resources needed to conduct subtasks inherent in the protocol. As with views, the telescience protocols must be matched to the agents' capabilities. Thus an end-to-end conceptual model of the integrating properties of a distributed system employing telescience operational modes is an important step towards designing information systems which tie shared world-views to the SAIS/SSIS/LSIS suite of services. The proposed model is also useful for determining the implications of partitioning operational intelligence between users, resources, agents and telescience services, particularly in the maintenance of the shared-world knowledge.

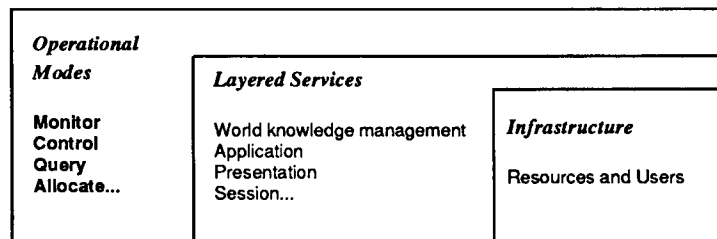
Model Overview

The conceptual shared-world model consists of distributed *active-elements* (*users* and *resources*); their intelligent cooperating representatives, *agents*; and *telescience protocols* for interacting which link the agents together into a communicating, cooperating whole capable of performing operational *tasks* and maintaining *shared-world* knowledge. Agent functions and telescience protocols are summarized in Figure 1; Figure 2 diagrams the model at a high level for the Space Station operational environment; definitions follow.

Figure 1
Agent Functionality and Telescience
Protocol Overview



Telescience Protocol Attributes



Active-elements: Examples of Space Station *users* are: crew, ground-based operators (e.g., at the Discipline Operations Control Center), principal investigators, and commercial clients. *Resources* include discipline independent entities such as the Space Station Information System and inventory systems which control the allocation of consumable materials such as food, water and electrical power, and discipline resources such as the Life Sciences centrifuge and live experimental subjects. Users and resources are termed active-elements since they may initiate actions, accept and produce information and feedback, or undergo changes in physical state. It is useful to distinguish two classes of active elements: those comprising infrastructure (e.g., a knowledge base of station-wide power resources and allocation constraints) and those outside the infrastructure (a self-regulating habitat airflow subsystem) since the former impacts the world knowledge base much more than the latter.

Agents: Interactions between active elements in Space Station scenarios are almost always mediated by technologies such as on-board information systems and their user interfaces, video links, teleoperations workstations, etc. There

are also internal idiosyncratic mediators: skill levels and other cognitive factors, or machine/machine interfaces (e.g. a humidity sensor/data acquisition interface). The collective functions and properties of these mediators are assigned to *agents* in this model. It is our contention that these agents must exhibit a degree of intelligence, since an agent acts as a standardized operator which represents, interprets and informs its active element (i.e., constructs a view of the shared-world for the active element), as shown in Figure 1. A group agent represents a collection of agents (see Figure 2); this arrangement may be hierarchical. Agents develop increasingly complex roles to reflect enhanced capabilities, responsibilities and interdependencies of their associated active elements and linkages to other agents in the system.

Telescience Protocols: Telescience protocols structure and govern the behavior of agents as they bind to perform tasks. Examples of Space Station operational modes employing telescience are planning, scheduling, control and monitoring. Examples of telescience service protocols are resource allocation, transaction management and telecommanding. Telescience relies on a base infrastructure, including communication networks and human teleoperators, to obtain infrastructure resources needed to conduct subtasks inherent in the protocol. Telescience protocols can be considered a set of layers, each providing services (presentation, application, etc); the upper layers maintain the world information base and present appropriate shared-world views to agents.

Tasks: Representative distributed Space Station tasks relying on a shared-world model include revising an experiment protocol, remotely operating a tool, advising a crew member, or obtaining a reading from an experimental instrument. Tasks are initiated by active elements and performed through agents. Tasks require the identification of necessary agents, establishment of a telescience protocol of cooperative behavior via selection of a set of services, the exchange of information and feedback, etc.

Maintaining the Shared-World View: Agents must be cognizant of their own actions and the effects of actions undertaken by other agents. Their world's integrating properties must persist and evolve as its elements are replaced, enhanced, and become more interdependent. A shared-world knowledge base (or bases, it need not be centralized) and rules embody this knowledge. During the execution of a task, telescience protocols record each agent's contribution to the shared-world by maintaining the world knowledge base, a sort of blackboard. For example, one can initially think of a telescience protocol as a kind of schema, or script, with roles assumed by the agents. A planning activity undertaken by two agents might then result in relevant information being posted to the world knowledge base at the conclusion of the planning task; the telescience mode governing the planning interaction would determine the presentation and content of this information. A more powerful conception than a schema is the incorporation of a dynamic model (e.g. a mental model) into the telescience protocol. The transient model would represent a particular unique operational situation and the expectations that flow from it; condition-action rules could be used to flexibly construct and interrelate the shared-world knowledge and to carry out procedures. Predictions about the attainment of goals would be the major source of feedback.

Life Sciences Considerations

Performing life sciences research on board the Space Station will require that the operational environment accommodate a number of needs and constraints which are specific to life sciences [7,8]:

- a high degree of uncertainty: science involving living systems is inherently variable and predictive models are often inadequate;
- accommodation of "art" and technique in administering experimental treatments and subjectivity in interpretation of data;
- support of frequent real-time interactions with experiments (both manual and automated);
- a high level of crew expertise and PI involvement;
- labor-intensive experiment maintenance;
- ground-truth correlative studies;
- relatively long duration experiments.

The model proposed here presents conceptual and implementation choices which are difficult to resolve a priori. The authors believe the nature of the scientific discipline and its accustomed operational environment (e.g. SpaceLab experiences) will influence answers to questions such as the following:

- How does the agent learn about, interface with and represent its active element? What cognitive loads are reasonable for an agent to impose on a user?
- How much variation between similar agents is permissible and how much standardization can be imposed before system functionality is perceptably degraded? Do "similar agents" imply similar active elements?
- Should telescience protocols and services provide session-based negotiation mechanisms (e.g. for a needed resource) or should agents utilize global transaction managers which are independent of the telescience protocol?
- How do semi-autonomous agents request assistance from active elements? How do active elements understand and respond to an autonomous agent?

The nature of Space Station life sciences research and operations illustrates the need for discipline-specific agent capabilities and telescience protocols. In the following sections, three distributed task interaction types, illustrated through prototypical Space Station life sciences scenarios, are used to explore such questions.

Modelling Distributed Task Interaction Types

Use of the term *distributed task* implies here that the active elements involved are typically separated in space and (importantly) time so that telescience operational modes must be invoked and agents must assist. Three task interaction types will be discussed below:

- user/user task

(illustration: teleconferenced collaboration between 2 primary agents)

- user/resource task

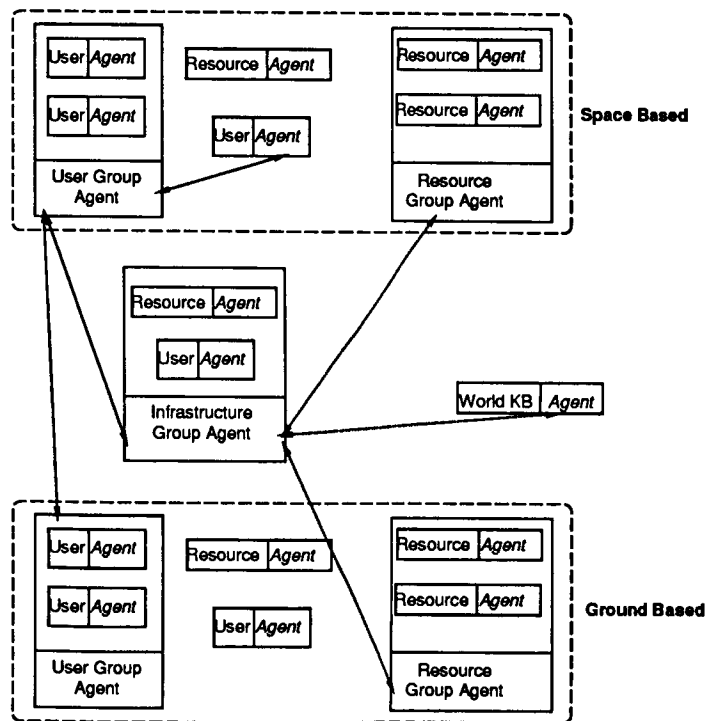
(illustration: habitat monitoring and control between 2 primary agents; many secondary information system agents are involved.)

- user/user/resource task

(illustration: PI assisting crew use of telerobotics; three 2-way interactions, one 3-way interaction; many secondary information system agents are involved).

The interaction combinatorics increase exponentially with the number of active elements. As the telescience protocols become correspondingly richer, and the world increasingly complicated, agents require ever more refined shared-world views and dynamic modeling capabilities.

Figure 2 Distributed Model for Space Station Operations



Links represent task-dependent telescience protocols governing interacting agents

Agents cooperating on a task share a common telescience protocol (two such tasks depicted here)

User/User Distributed Task

A typical (distributed) user/user task is **teleconferenced collaboration** between a crew member and ground based scientist, arising, for example, from an unexpected experiment result. With only a few participants in a conventional setting, conferencing humans themselves perform most of the functions needed, including dialog management, establishing a protocol for negotiation, selecting the appropriate communications media, and obtaining or recalling knowledge needed for decision making, etc. [9]. However, distributed user/user collaboration for Space Station life sciences will frequently if not typically involve numerous participants (as many as 30 PI's are anticipated for some synchronous sets of experiments) who will need to discuss and negotiate the allocation and configuration of numerous experimental resources, both ground and spaced based, on numerous occasions over long time periods. The actions of these users will then affect other Space Station users and resources.

Informal coordination of such a complex process far exceeds the cognitive capacity that the group can allocate to keeping the process on track. An explicit model -- a shared-world view -- is needed to facilitate user/user tasking, render the operating context comprehensible and inform the world of changes in active elements resulting from carrying out the joint task. In the scenario for example, the experiment PI will require and be capable of processing more experiment-specific information than the less experienced crew member. Thus many of the functions performed by humans in simple conversational interactions must be migrated to intelligent agents which understand, enforce and refresh the shared-world view. Some of the needed model enhancements are enumerated below.

- *Active elements:* resources (semi-automated) to coordinate user-to-user session arrangement; knowledge bases to support decision making; knowledge bases for recording decisions and forecasting consequences.
- *Agent capabilities:* knowledge base access; rule-based augmentation of behavior; multimedia dialogue support; collaboration; negotiation.
- *Telescience protocols:* real-time multi-media teleconferencing management; collaboration and negotiation services; updating of "history" in an archival world representation or knowledge base.
- *World knowledge base elements:* collaboration participants, results, forecasted consequences, affected active elements and agents.

Even for the relatively simple task of teleconferencing, maintaining a shared-world view and structuring the interactions requires considerable embedded intelligence in the system's agents and telescience infrastructure.

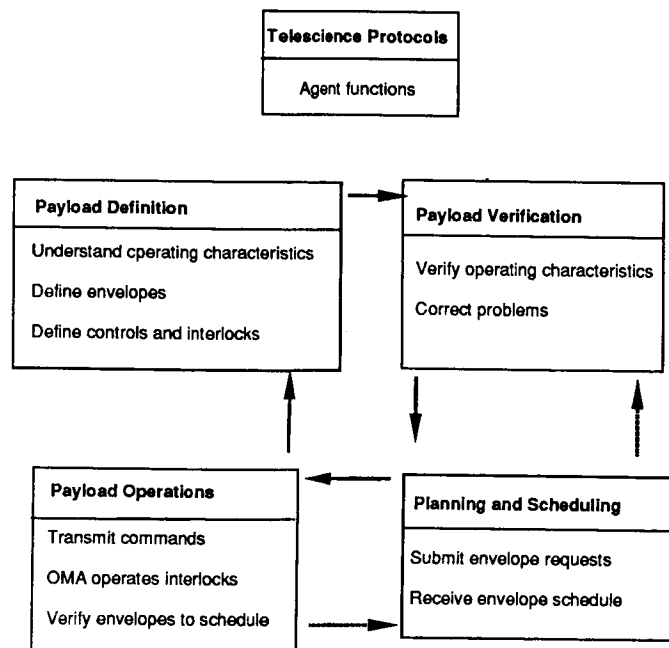
User/Resource Distributed Task

A typical distributed task in life sciences operations is **remote monitoring and control** of a life sciences payload experiment parameter (e.g. some biomedical factor) by a remote expert [10]. Life science experiments, because of their unpredictability, will generally require more frequent and skilled interaction than other discipline experiments. While there are two primary

agents (the remote user and the payload sensor), many secondary agents are involved, including numerous SSIS and payload specific information system elements. For example, a user agent for the remote expert might reason about the current operational envelope and assist the expert by establishing a context for operations activity on a teleoperations workstation. Such assistance, could automatically make available relevant DBMS's and expert systems. One of the very important system capability which emerges from this scenario is the need for transaction management to control distributed access to the experiment resource.

Transaction management : The objective of transaction management is to control the effect of transactions (not the transactions themselves) on payloads, thereby ensuring safety and preventing potential interference [11, 12]. Transaction management is supported by the SSIS Operations Management System and discipline payload control systems. The PI and other users initially define the payload operational *envelopes*, which are sets of required resource consumptions, environmental requirements and impacts. The payload's operation is characterized as a time sequence of its operational envelopes. A payload definition is needed to define controls and interlocks, but a user is free to control the operation of a payload by sending transactions as long as their effects do not violate the operational envelopes. Planning and scheduling subsystems are needed to submit operating envelope requests and receive envelope schedules; payload operations subsystems format and transmit commands; and the SSIS Operations Management Application operates interlocks and verifies the envelopes are consistent with the schedule. Figure 3 illustrates transaction management in accordance with the proposed model. Since transaction management is an integral part of telepresence, it is perhaps best embedded in the telepresence protocol (e.g. teleoperation) rather than as an agent function.

Figure 3 Transaction Management



User/User/Resource Distributed Task

A more complex distributed task typical of life sciences operations is a scenario involving a ground based principal investigator who is interacting with a crew member to perform an experiment protocol involving a semi-autonomous resource, such as telerobotics. The crew person will generally direct and monitor the robot locally, though the remote PI may wish to occasionally intervene or instruct. The real-time requirements and interaction complexities require rapid exchange of shared-world data: for example, near-real time video images of the robot movements.

Implications of Autonomy in a Telescience Context Implementing autonomous operations at various hierarchical levels is intuitively appealing to Space Station facility designers, including life scientists, as they attempt to make scarce resources such as crew time go farther by introducing automated subsystems. This perspective raises some concerns, however, when the motivation for autonomy is related to telescience and teleoperation:

- How can autonomy of one active element or group be reconciled with human-in-the-loop telescience modes?
- What features must autonomous operation of a subsystem embody and project in order to assure the rest of the system -- including users -- that the autonomous subsystem is performing adequately? [13]

On the first point, the problem of mixing autonomy with flexibility is important. Often, for example, the human user may wish to "override" autonomy temporarily; the system must enable a smooth transition and restore autonomy after the interrupt. The operational environment must provide for the autonomous component to share fully in the common world-view so that there are no "memory gaps". Such overhead can cut into the cost savings achievable through autonomy in the first place, so that telescience may dictate fewer high-level autonomous subsystems.

On the second point, which is a species of the reliability question, autonomous agents may need to have special agent features to enable the rest of the system to recognize them as such and act accordingly. This may especially apply to transactions between autonomous components and live users, from the standpoint of assuring user confidence in the autonomous element's performance quality. Numerous studies in the human factors literature show the need for humans-in-the-system to be informed frequently that autonomous processes running "behind the scenes" are normal; otherwise the operator is likely to perturb the system to investigate, which may degrade overall performance.

Testbedding to Explore Model Concepts

The ARC Life Sciences Telescience Testbed is currently under development to evaluate the application of telescience to Space Station Life Sciences. The roles of the testbed active elements, agents and telescience modes will be explored by conducting operational scenarios (e.g. seed planting) with

References

- [1] Leiner, Barry and Weiss, James, Telescience Testbedding: An Implementation Approach, RIACS TR 88.2, 1988.
- [2] Space Station Program Office, Johnson Space Center, Space Station Information System Architecture Definition Document, JSC 3025 Rev. A, January 15, 1987.
- [3] P. Shames, Toward a Science Data System Architecture, Draft, SAIS Science Data Management Panel Report 3-6.
- [4] Winograd, Terry and Flores Fernando, Understanding Computers and Cognition, Ablex Publishing, 1986.
- [5] Holland, John, et. al., Induction, MIT Press, 1986.
- [6] Norman, Donald, Some Observations on Mental Models, Mental Models, Gentner, Dedre, editor, Lawrence Erlbaum Publishers, 1983.
- [7] Arno, Roger, Accommodating Life Sciences on the Space Station, Paper 871412 in Proceedings of the Seventeenth Intersociety Conference on Environmental Systems, Seattle Washington, July 1987.
- [8] D. Rasmussen, Life Sciences Research Facility Automation Requirements and Concepts for the Space Station, Paper No. 860970 in Proceedings of the Sixteenth Intersociety Conference on Environmental Systems, San Diego, CA, July 1986, p. 539.
- [9] Winograd, Terry, A Language/Action Perspective on the Design of Cooperative Work, Stanford University Report STAN-CS-87-1158, 1987.
- [10] Committee on Space Biology of Medicine, Strategy for Space Biology and Medical Science for the 1980s and 1990s, National Academy Press, 1987
- [11] Space Station Information System Concept Document, February 1988.
- [12] T. Secord, Life Sciences Facility Control and Telescience Systems, McDonnell Douglas MC3658, draft, March 1988.
- [13] Michalowski, Stefan, Rehabilitation Research and Development Center, Stanford University, personal communication.
- [14] D. Rasmussen, A. Mian, J. Bosley, Telescience Testbedding for Life Science Missions on the Space Station, presented to the 26th AIAA Aerospace Science Meeting, Reno, Nevada, January 13, 1988.

N88 - 30334

Artificial Intelligence in a Mission Operations and
Satellite Test Environment

Carl Busse, NASA Jet Propulsion Laboratory

A Generic Mission Operations System using Expert System technology to demonstrate the potential of Artificial Intelligence (AI) automated monitor and control functions in a Mission Operations and Satellite Test environment will be developed at the National Aeronautics and Space Administration (NASA) Jet Propulsion Laboratory (JPL). Expert System techniques in a real time operations environment are being studied and applied to science and engineering data processing. Advanced decommutation schemes and intelligent display technology will be examined to develop imaginative improvements in rapid interpretation and distribution of information. The Generic Payload Operations Control Center (GPOCC) will demonstrate improved data handling accuracy, flexibility, and responsiveness in a complex mission environment. The ultimate goal is to automate repetitious mission operations, instrument, and satellite test functions by the application of Expert System technology and Artificial Intelligence resources and to enhance the level of man-machine sophistication.

The work described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Introduction

The NASA Jet Propulsion Laboratory will provide the functional requirements, conceptual design, and interface definition, detailed design, software coding, and unit testing for the development of a Generic Payload Operations Control Center (GPOCC). The GPOCC will apply Artificial Intelligence (AI) to support of instrument and satellite test environment, as well as Mission Operations. The GPOCC will couple current Expert System (ES) developments with developments in computer display technology and intelligent man-machine interface features to develop the imaginative improvements necessary for rapid data interpretation. Adaptive "Smart card" Input/Output (I/O) ports will be used for external communications blocking and deblocking.

Expert System technology will be applied to four distinct areas:

- o Mission Flight Planning
 - Flight & Instrument Sequence Planning
 - Command Constraint Checking
 - Mission Rules
 - Flight Rules
 - Spacecraft Status
 - Instrument Status
 - Ground Constraints
 - Orbit Cycle Activity Profile
 - Sequence of Event Generation
- o GPOCC Control of On-Board Data Management
 - NASA Standard Tape Recorder
 - NASA Standard Spacecraft Computer
 - Memory Management
 - Memory Comparison
 - Instrument Memory Management
 - Memory Management
 - Memory Comparison
- o DSN 26 meter subnet and TDRSS Telecommunications Scheduling
- o Satellite Telemetry Data Monitoring, Trend Analysis, Prediction Forecast, Anomaly Detection, Fault Identification, Diagnosis, and Correction Action Strategy

The Generic Payload Operations Control Center effort is intended to support the demanding transitions between instrument and satellite development, integration and test and flight operations of many classes of Earth orbiting payload.

The Role of the Jet Propulsion Laboratory

The development of an operational Expert System based command and control system to provide Mission Operations and Satellite Integration support is directly applicable to JPL's NASA Mission. This implementation also applies directly to the JPL Mission Operations Productivity Enhancement Program (MOPEP) in support of the Voyager spacecraft's upcoming Uranus Encounter and the application of Expert System Assistance to aid the Galileo Spacecraft integration effort.

In an era of extremely limited human resources the use of the latest technology is mandatory. Expert systems are ideally suited to appropriately defined and baselined mission operations as well as instrument and satellite test environments. Artificial Intelligence technology is rapidly becoming the technological leading edge of new "User Friendly" systems. 1

The development of a Generic Payload Operations Control Center at the Laboratory may lead to the development and flight of a new suite of JPL instruments integrated and supported from the JPL developed Generic Payload Operations Control Centers.

The Jet Propulsion Laboratory will provide project management for the development of the Generic Payload Operations Control Center. Detailed GPOCC Functional and Software Requirements will be generated prior to prototype implementation. JPL will also be responsible for integration and system acceptance testing.

Applicability

The application of expert systems in support of NASA, and Department of Defense (DOD), as well as, the National Atmospheric and Oceanographic Administration (NOAA) Earth observation missions is a significant step in the acceptance of Artificial Intelligence.

The Generic Payload Operations Control Center concept could support a variety of NASA and DOD missions, including Scout Explorer instruments and payload, Low Earth Orbit missions such as Shuttle launched Free Flyers and Get Away Specials, earth observation satellites typified by Landsat, Quick Sat, as well as enhancing a variety of other possible missions. These potent applications include a small, highly portable, and survivable secure Command and Control center for DOD missions. With the addition of the NASA Data Link Module (NDLM) which provides a direct forward and return services to the NASA Tracking and Data Relay Satellite (TDRSS) the GPOCC provides direct TDRSS forward and return links. Also with the addition of a 9 meter antenna, antenna servo drive and Radio Frequency equipment racks, the GPOCC becomes a miniature and easily transportable integrated tracking and ground data system.

The GPOCC provides significant benefits in areas where budget, mobility requirements, and manpower limitations restrict the type of ground process available to project and science team members. The GPOCC allows the advances in Artificial Intelligence (AI) technology in the past fifteen years to be applied to a "real world" mission environment. These technological areas reduce the impact of limited experienced human resources. 2

The use of expert systems in the GPOCC permit improved control of the decision trees such as the application of flight rules to the spacecraft command generation process. The challenge will be to apply these systems to the control of a large and dynamic knowledge base. The GPOCC is a solution for a wide range of missions, as well as the kernel on which to expand or adapt broader applications.

Instrument Test Environment

The GPOCC allows the instrument scientist to carefully observe instrument operating conditions during development in the same operations environment and equipment configuration as will be used in actual mission flight conditions.

Satellite Test Environment

The GPOCC adds a new dimension to the satellite and test environment. GPOCC data handling flexibility and the sophistication of integrated display technology lends itself to the task of releasing satellite integration and test personnel from constructing and preserving a complex data handling system.

Mission Operations Environment

The GPOCC is unique in that Mission operations support will be a natural progression from the instrument and satellite integration and test environment. The GPOCC support of Mission Operations System will differ little from the prelaunch Ground Data Systems testing and MOS training activities. The major difference is the addition of "live" tracking data input to the Mission Planning and Navigation process. This similarity significantly aids in the transition from single instrument integration to full up on orbit operations.

Generic Payload Operations Control Center (GPOCC)

The Generic Payload Operations Control Center (GPOCC) system is composed of five major elements:

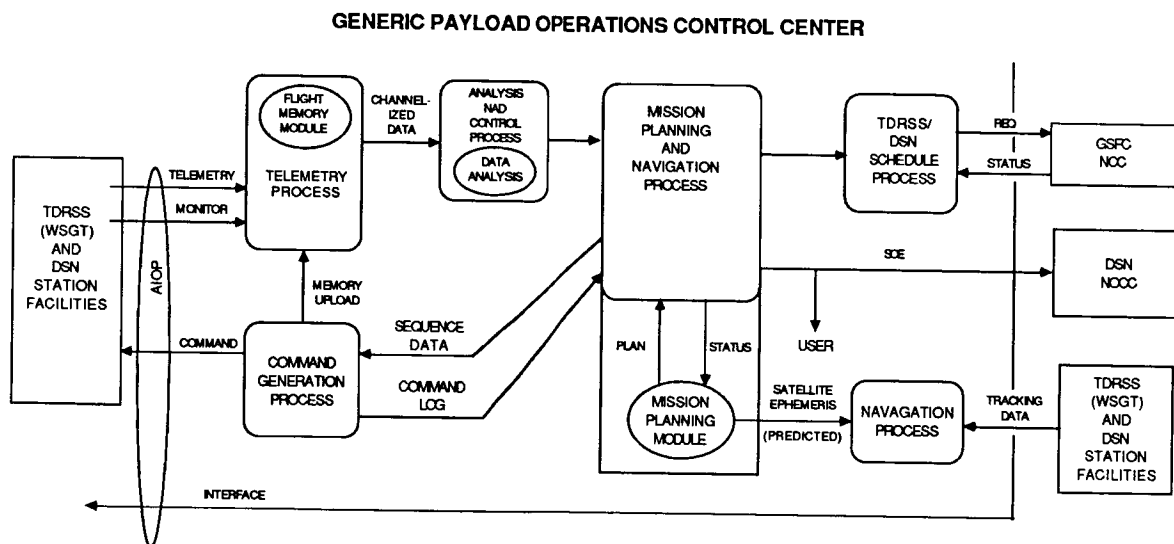
- 1) the telemetry processor
- 2) the command generation process
- 3) the Mission Planning and Navigation process
- 4) the TDRSS, Deep Space Network 26 meter subnet telecommunications scheduling process
- 5) the Analysis and Control process.

Five expert systems modules interface with these processes. These expert systems include the:

- 1) the Flight Memory Module (FMM)
- 2) the Data Analysis Module (DAM)
- 3) the Mission Planning Module (MPM)
- 4) the Mission Rules Module (MRM),
- 5) the Mission Scheduling Module (MSM).

The GPOCC telemetry, command and mission planning and navigation processes will also couple current Artificial Intelligence and expert systems developments with intelligent display technology to provide more user efficient data interpretation by use of symbolic representation including interactive Icons.

The overall GPOCC System is functionally represented in the following diagram.

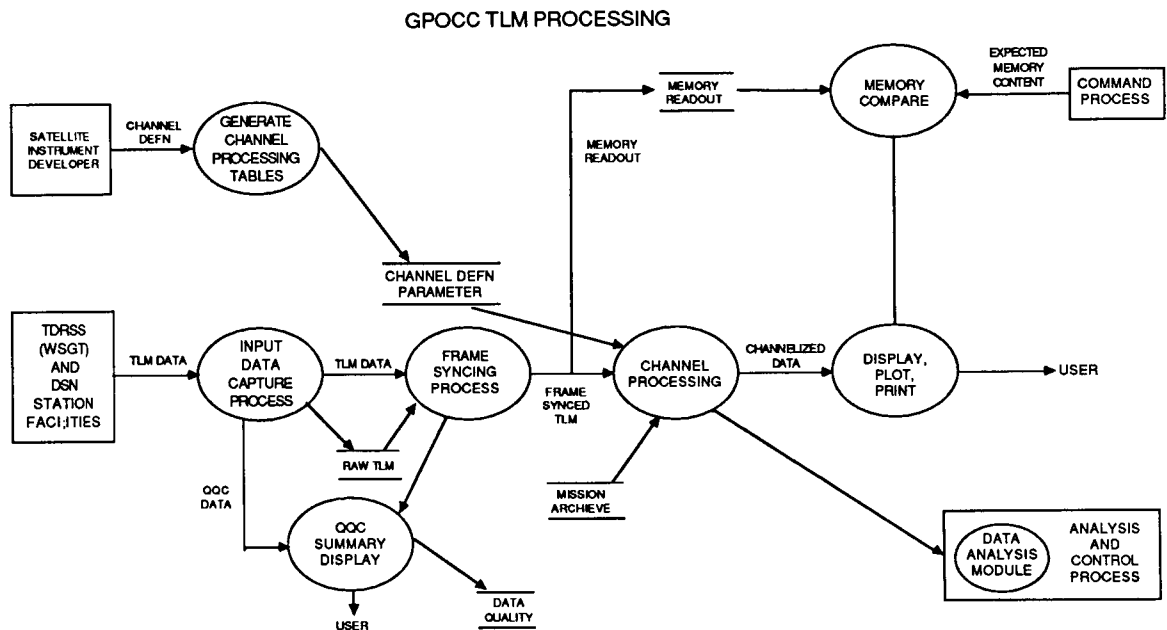


Telemetry Process

The GPOCC telemetry process provides the GPOCC with communications links to the Tracking and Data Relay Satellite System (TDRSS), White Sands Ground Terminal (WSGT) and the Jet Propulsion Laboratory's Deep Space Network (DSN) 26 meter subnet. NASCOM and DSN Goddard Interchange Blocks (DGIB) are deblocked via an Adaptive Input/Output Port (AIOP) utilizing a single 256 KB Random Access Memory (RAM) smart card per I/O port. Telemetry data is frame synchronized, decommutated and channelized, and archived on optical disks. Multilevel limit checking, and dynamic alarms are used to alert the user of data value irregularities.

The telemetry process contains the expert systems Flight Memory Module (FMM). Channelized data is passed to the Flight Memory Module and the Analysis and Control process through a seamless interface which is invisible to the user. The Flight Memory Module (FMM) tracks the status of the payload recording device and inputs into the Mission Planning process record, erase, and data playback recommendations.

GPOCC Telemetry processing is presented in the accompanying figure.



Flight Memory Module (FMM)

The FMM receives input from the telemetry stream following frame synchronization and data channelization. The FMM will determine which tape recorder is in use, the data stored on each recorder, the track being recorded, and the proximity to tape recorder End-of-File (EOF). The output of the FMM will be recorder cycle statistics, data volume, time boundaries, T/R playback and erase schedule requests to the Mission Planning Process.

The FMM will also contain memory maps for the satellite On-Board Computer (OBC), and science instrument memories. The FMM performs a comparison of memory load variables anticipated by command sequence inputs from the command process against current memory values derived from the downlink telemetry process. If discrepancies are detected, command requests are passed to the Mission Planning Process. Memory variable values are continually checked against forecasted limits to guard against Single Event Upsets (SEU), and memory failures. Statistics of memory usage are calculated to determine memory management strategies.

Analysis and Control Process

The Analysis and Control Process provides data analysis, satellite anomaly detection, fault identification, and corrective action as well as fallback strategy. This process also provides data plotting, tabulation, and trend analysis. The process records data metrics for satellite and ground system performance evaluation.

The Data Analysis Module Expert System resides in the Analysis and Control Process.

Data Analysis Module (DAM)

The Data Anomaly and Analysis Module monitors data values to detect event and trend variations for analysis of fault conditions and recommended corrective action. The DAM receives input from the telemetry process and performs continual data monitoring and trend analysis based on historic data archived on the telemetry process optical storage device. Faults are identified, a diagnosis is preformed, and corrective action strategy proposed. 3

This Data Analysis Module will monitor telemetry data and perform continual data monitoring and trend analysis based on its knowledge base and historic data archived on an optical disk storage device. The system maintains a continuous "knowledge" database of past system performance characteristics.

The Data Analysis Module will be partitioned into four stages:

- | | |
|----------------|--|
| MONITORING | - monitoring, and interpreting, instrument and satellite behavior. |
| DIAGNOSIS | - determine origin of system malfunctions inferred from knowledge base. |
| PREDICTION | - inference of predicted performance based on historic performance and current trends. |
| RECOMMENDATION | - developing and prescribing corrective action for diagnosed problems. |

Command Generation Process

The GPOCC Command generation process accepts command sequences from the Mission Planning and Navigation Process. The GPOCC command configuration allows realtime commands to be issued from the mission data & command workstation display console. The realtime command display allow for selection of immediate execution, and timed commands to be stored in the instrument and satellite memories. The Command process echoes realtime commands to the Mission Planning and Navigation process for cross-checking

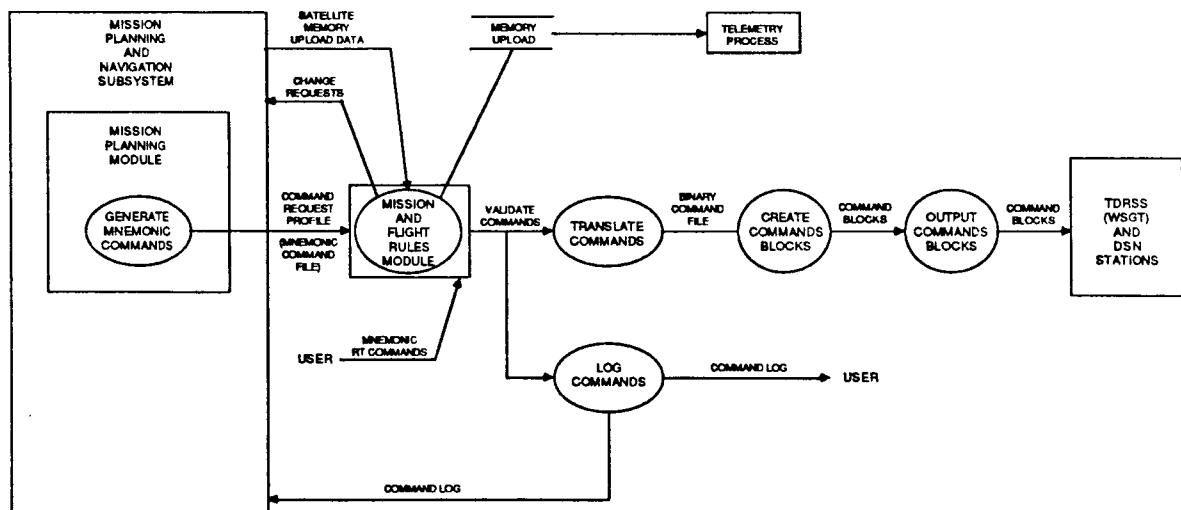
commands in the Mission and Flight Rules Module. Verified MP&N and realtime commands can be transmitted automatically, or on manual user request. All commands are cross referenced against the current configuration of the spacecraft (from the telemetry processor) to verify consistency with Mission flight rules, perform resource conflict resolution, and insure state dependent command compliance. Transmitted commands are recorded in a Command Activity Archive.

In the event of a satellite anomaly, the command Generation process accesses stored contingency procedures and emergency command sequences for rapid transmission to the satellite. The recommendation for the contingency and emergency command sequences are part of the correction action as suggested by the Analysis and Control Process Data Analysis Module.

The Mission Rules Expert System resides in the command process.

The command generation process data flow is indicated by the following diagram.

GPOCC COMMAND GENERATION PROCESS DATA FLOW



Mission Rules Module (MRM)

Command sequences created as part of the Mission Planning process, as well as realtime commands, are checked for compliance with mission and flight rules in the Expert System Mission Rules Module (MRM). Commands are also checked against current and

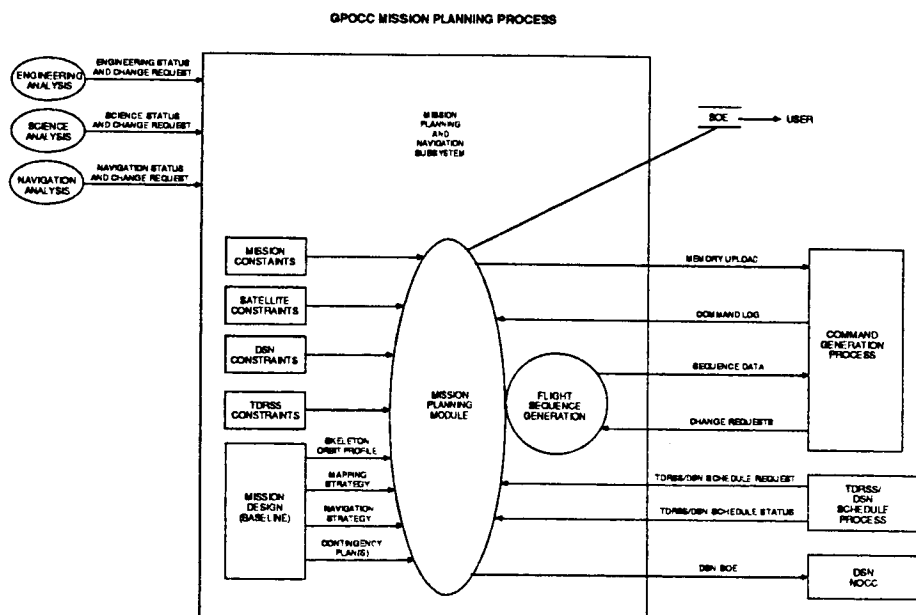
expected satellite and instrument states, and the ground station constraints imposed by the Deep Space Network 26 meter stations and the Tracking and Data Relay Satellite System. The MRM also resolves conflicts for satellite and sensor resources. Following evaluation of the verified commands, the command file, composed of command mnemonics is translated into a binary commands and blocked in the Adaptive I/O Port and transmitted to the DSN 26 meter subnet stations or TDRSS White Sands Ground Terminal for transmission. Commands which fail compliance checks are flagged to the command operator and transmission denied. A reject override will be provided. Command sequences are returned to the Mission Planning Process for regeneration. The command - mission planning interface is seamless so that the operation and location to the MRM is invisible to the User.

Mission Planning and Navigation Process

The Mission Planning and Navigation process provides the GPOCC with a daily mission profile, satellite orbital parameters, satellite geographic position (from GSFC tracking data) and attitude (from the telemetry processor), and command sequence generation. The Mission Profile is provided the Telecommunications Scheduling Process for TDRSS and DSN support scheduling.

The Mission Planning and Navigation Process creates command sequences to meet science and mission requirements. Sequences are passed to the command process.

The Mission Planning and Navigation process is shown in the following diagram.



Mission Planning Module (MPM)

The MPM creates individual orbital profiles from a static profile skeleton and generates command sequences based on Mission and Instrument for each orbit cycle. Instrument and satellite activities are scheduled in detail. Much of the work on the Mission Planning Module will be based on work previously done at JPL by the Artificial Intelligence and Mission Planning groups. 4

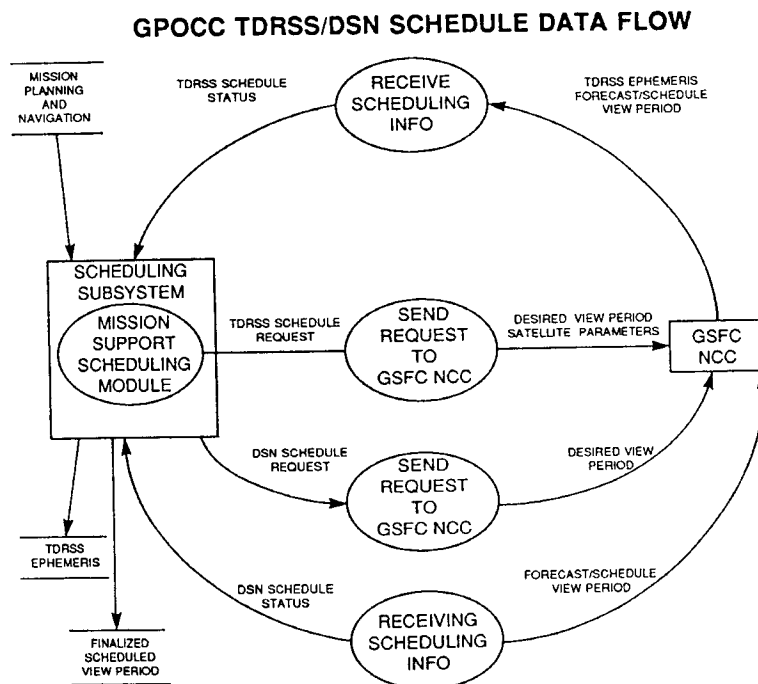
Telecommunications Scheduling Process

The Scheduling Process receives Mission Profile input from the Mission Planning and Navigation Process. Using the Mission Profile the expert systems Mission Scheduling Module generates DSN 26 meter subnet, TDRSS, and communications line support requests. These support requests are forwarded to the Goddard Space Flight Center Network Control Center (NCC) for confirmation. Schedule confirmation messages are received by the GPOCC Scheduling process for acknowledgement or modification.

Mission Scheduling Module (MSM)

The MSM receives input from the Mission Planning and Navigation process. The MSSM is interactive with both the DSN 26 meter scheduling system, the TDRSS Network Control Center (NCC) at the Goddard Space Flight Center (GSFC). Schedule requests which cannot be supported are reported back to the Mission Planning and Navigation Process for revision of the orbit cycle activity profile.

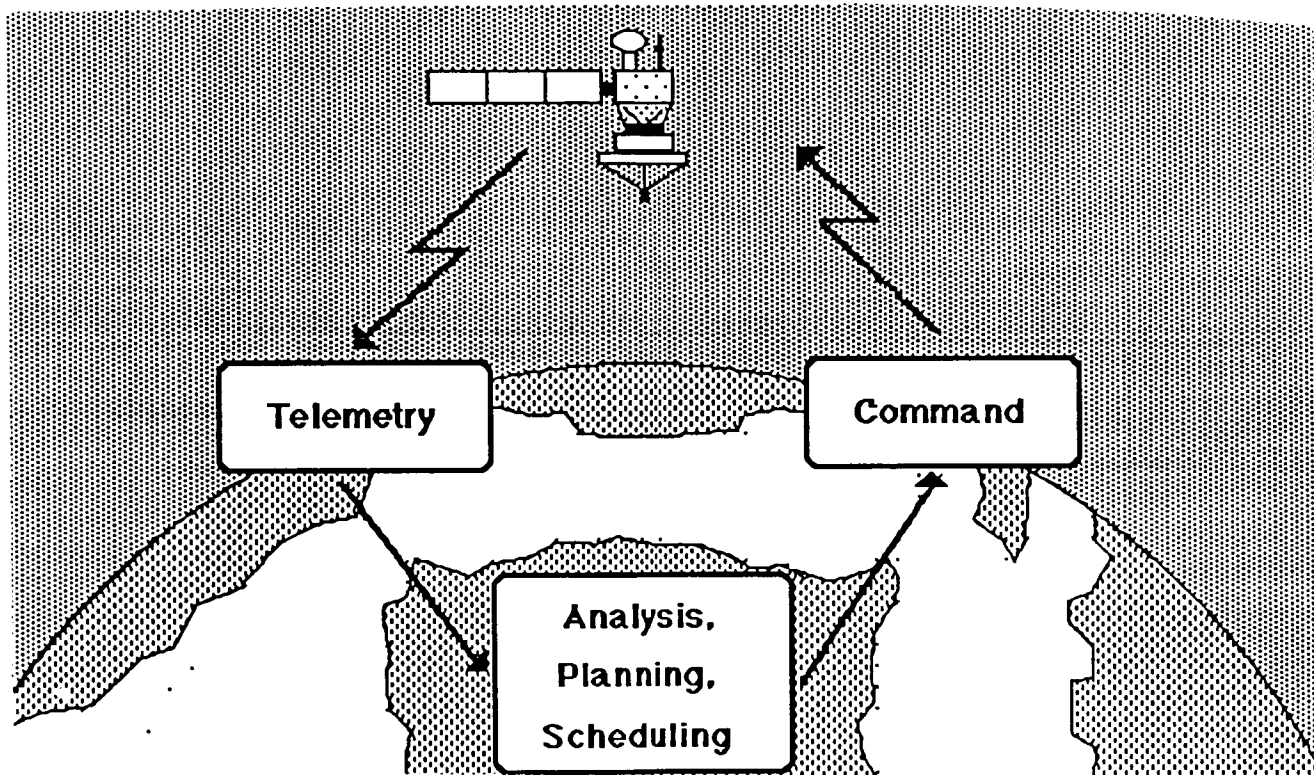
The GPOCC TDRSS/DSN schedule data flow is shown in the following diagram.



Display Technology

The GPOCC system man-machine interface will be via easily interpreted high content graphic displays. These displays will provide a direct representation of the intrinsic images associated with the telemetry, command, telecommunications, and mission planning and navigation processes, as well as the instrument and satellite systems. Multiple displays screens will be linked through context and mouse sensitive icons and text.

An example of the top level GPOCC interactive display showing the satellite systems, telemetry, command, and Mission Planning and Navigation process icons is shown below.



Implementation

The GPOCC System design policy will be to insure the GPOCC implementation meets the requirements specified in JPL Software Standard D-4000, as well as appropriate JPL Level II Software Standards. 5

The implementation Generic Payload Operations Control Center will consist of a phased development program. The goal of this incremental development approach is to allow a phased step by step development of elements. The initial implementation will be the Adaptive I/O Port, telemetry process and the Flight Memory Module (FMM), and the Data Analysis Module (DAM). This front-end telemetry development portion is necessary for further GPOCC development, and to support instrument and satellite integration and test.

The following delivery will consist of the command and telecommunications processes, exclusive of their associate expert systems, the Mission Rules Module and the Mission Scheduling Module.

The most difficult phase, Mission Planning and Navigation process will be the final delivery. This implementation will include the Mission Planning Module, and Mission Scheduling Module. Halfway through this implement, development of the Mission Rules Module and Mission Scheduling Module will be completed and integrated into the command and scheduling processes.

This phased implementation will allow early confirmation of data system integrity and compatibility through the use of top-down design for the non-expert systems portion of the GPOCC. Coding will be prioritized and sequential testing will be used to insure that coding cannot proceed until the system requirements are well understood, documented, approved and that the preceding code is validated. The detailed design, coding, debugging, unit testing, and integration of each increment will be performed sequentially allowing the results to be fed back into subsequent builds.

The GPOCC expert system modules design from the bottom-up to allow a carefully understood implementation. The completed modules will then be integrated will the GPOCC processors.

Over all regression testing will insure preceding development is not impacted by subsequent software builds.

This methodology will also allow time for system quality assurance and software documentation to keep pace with code development. Phased implementation will permit early software transfer to unit and user acceptance testing providing timely feedback to the development group, and better understanding of system capabilities by system user community.

Final User Acceptance Testing will insure compliance will overall GPOCC system and specific user mission requirements.

Conclusion

The expert systems in the Generic Payload Operations Control Center provides for consistent, dependable and validatable performance, will demonstrate thorough and reliable and fast reasoning, and to greatly reduce the requirements for a sizable test and mission support staff.

ACKNOWLEDGMENTS

The author gratefully acknowledges the guidance and suggestions from Mr. David Klemp, who provided the genesis of the project, Dr. Anil Agrawal, Mr. Harry Avant, Mr. Jackie Giuliano, Dr. James Willett, Ms. Suzanne Sellers, and Mr. Willems of the Jet Propulsion Laboratory; Ms. Betty Sword of Federal Electric Corporation; Mr. Larry Shelley, Mr. Max Kostiner, and Ms. Diana Berry of Computer Sciences Corporation; Dr. Rogers Saxon, of CypherMaster; Mr. Walter Gonzalez and Mr. Keith Stuart of Innovative Information Systems; and Mr. Robert Bartlett of Fairchild Space Company. This paper could not be prepared without the tireless assistance of Ms. Jean Iannitti, Ms. Susan Lineaweaver, Mr. Jim Ingles and Mr. Warren Moore of JPL.

REFERENCES

1. E. Hansen, "Lowering the Cost of Satellite Operations", American Institute of Aeronautics and Astronautics, AIAA-88-0549, (1988).
2. JPL D-5435, Generic Payload Operations Control Center Function Requirements Document, (1988).
3. P. Harmon ed., "Expert System Tools", Expert Systems Strategies, (1987).
4. D.F. Finnerty, J. Martin, and P.E. Doms, "Asset: An Application in Mission Automation for Science Planning", Journal of the British Interplanetary Society, (1987).
5. JPL D-4000, JPL Software Management Standards, (1988).

Automated Space Vehicle Control for Rendezvous Proximity Operations

Robert N. Lea

NASA/Johnson Space Center, Houston, Texas

ABSTRACT

Rendezvous during the unmanned space exploration missions, such as a Mars Rover/Sample Return will require a completely automatic system from liftoff to docking. A conceptual design of an automated rendezvous, proximity operations, and docking system is being implemented and validated at the Johnson Space Center (JSC).

The emphasis of this report is on the progress of the development and testing of a prototype system for control of the rendezvous vehicle during proximity operations that is currently being developed at the JSC. Fuzzy sets are used to model the human capability of common sense reasoning in decision making tasks and such models are integrated with expert systems and engineering control system technology to create a system that performs comparably to a manned system.

PRECEDING PAGE BLANK NOT FILMED

INTRODUCTION

Studies using fuzzy sets in modeling human common sense reasoning in decision making and applications to control processes such as sensor data editing and state vector update management in space operations have strongly indicated the utility of such methods.

In particular, fuzzy sets have been used for pre-editing star tracker data and controlling the processing of sensor data in prototype simulations of shuttle rendezvous. They have performed, independently of crew interactions, as well as the onboard system does, complete with the crew performing their functions of pre-editing data prior to processing. They have also been used to adequately perform the task of monitoring residuals during processing of data to guard against cases where there are unexpected problems that arise during the measurement processing time segment [1].

In further studies [2], it has been seen that fuzzy sets can be used to model crew actions in control of the shuttle during proximity operations. For example, if a small error in closing rate exists a small correction would be made in range rate. Furthermore, the model is done in such a way that rates and position in the Crew Optical Alignment Sighting (COAS) device are monitored continually so that if one of the desired conditions begins to degrade action can be taken to correct the condition before it becomes critical.

The reason the previously discussed study was undertaken was to create a system on which engineering studies that require a man in the loop could be done in a much quicker and less expensive way. The goal was to create a control system that reacted similarly to a pilot during rendezvous profiles. However, it became clear that if one can model a human flying the shuttle allowing this system to only process information that the crewman has available and in a way consistent with his ability to process information, one should be able to do an even better job if the system is allowed to process other relevant data that may not be available to the crew but is readily available to the system. Thus it seemed natural to consider these methods as applied to the problem of automated rendezvous.

AUTOMATED RENDEZVOUS VEHICLE CONTROL

The objectives of the automated rendezvous study are to create a set of software that will control the entire rendezvous sequence totally independent of human interaction. This study focuses on the proximity operations phase of such a mission. The previously referenced work on pilot modeling using fuzzy sets is applicable here as any automated vehicle control system should be able to perform the function of a human operator. In certain areas it seems clear that an automated system should be able to do a better job. The types of applications of fuzzy sets to piloting modeling that should be retained are models of decision making rules relating to the necessity of corrections and magnitude of such corrections to maintain a correct approach path.

Typical rules used for rendezvous vehicle control and modeled with fuzzy sets are the following.

If the rendezvous vehicles orientation with respect to a desired pointing vector to the target vehicle is close to the required orientation then no action is necessary.

If the orientation significantly deviates from the required then take appropriate action to correct the problem.

In the control system reported on here it was decided to use the π and S functions as given in [3] to model these rules since they are easily adjusted for varying degrees of "fuzziness" by varying the parameters that define their width and shape. The equations of the π and S functions are given below and their graphs are given in figure 1.

$$\begin{aligned}
 S(x,a,b,c) &= 0 && \text{for } x \leq a \\
 &= 2((x-a)/(c-a))^2 && \text{for } a < x < b \\
 &= 1 - 2((x-c)/(c-a))^2 && \text{for } b < x < c \\
 &= 1 && \text{for } x \geq c
 \end{aligned}$$

$$\begin{aligned}
 \pi(x,b,c) &= S(x, c-b, c-b/2, c) && \text{for } x < c \\
 &= 1 - S(x, c, c+b/2, c+b) && \text{for } x \geq c
 \end{aligned}$$

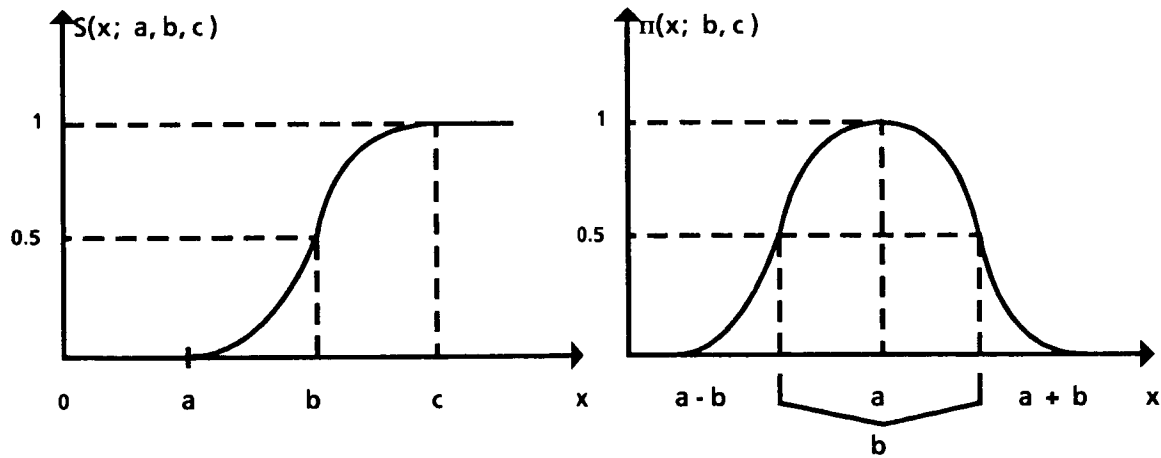


figure 1

Graphs of the functions used to model "significantly high", "significantly low" and "near" the desired position are given in figure 2. The desired position or state is labeled E in the diagram.

As can be seen one can effect a rapid or slow transision from complete membership to complete non-membership by altering the parameters a , b , and c or b and c for the S or π function respectively. Using these functions allow flexibility in the simulator for selecting a control strategy. Strategies can vary from the extremes of keeping the actual position and rates very close to their desired values, or only keeping the actual position and rates in some preset window of acceptable values. The way the functions are used will now be described.

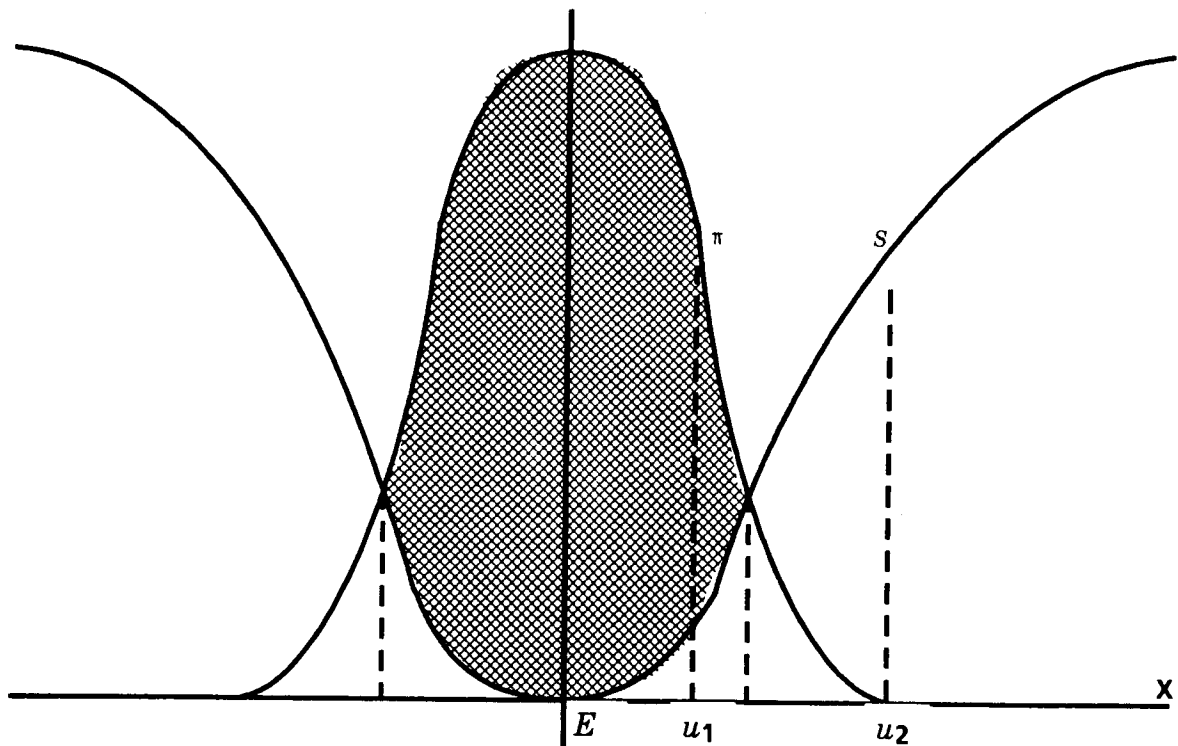


figure 2

Fuzzy sets are defined for "somewhat greater than", "somewhat less than", and "approximately equal to" the desired closing rate. They are also defined for "high", "low", and "near" with respect to the desired position. During some time interval (every two seconds for the shuttle) the fuzzy sets are evaluated and a determination is made as to whether an action needs to be taken to restore a rate or position to its desired value. If the no change function, such as "approximately equal to" or "near" the desired value, is larger than the corresponding change function, such as "somewhat greater than" or "low" with respect to the desired, then no action is taken. Otherwise an appropriate action is taken to restore the rate or position to the desired. The appropriate action is determined from an estimated action $A(u)$, where u is the current value of the state, required to restore the active vehicle to the desired position. This action $A(u)$ is then weighted by the change function $S(u)$ and an action $S(u) * A(u)$ is commanded to the system under control. Furthermore, there are no extreme accuracy requirements for the function $A(u)$. For example, referring to figure 2, if u_1 is the current value of x , then $\pi(u_1) > S(u_1)$ and no action is taken. On the other hand, if u_2 is the current value of x , then $S(u_2) > \pi(u_2)$ and an action $S(u_2) * A(u_2)$ is commanded. More than one action can be commanded at a time so long as a constraint of the system under control is not violated. For the shuttle the actions commanded are jet firings and are determined in the following way.

The required velocity change to effect a position change and/or an increase or decrease in range rate is divided by the proper setting of the digital autopilot (DAP). The DAP has two settings that are preloaded with values that control the magnitude of the jet firings. Typical values are 0.02 and 0.05 which translates into 0.02 or 0.05 feet per second change in velocity per pulse depending on which value has been selected. The nominal DAP setting is the larger of the two and is the proper setting if it is smaller than the required velocity change. If this setting exceeds the required velocity change then the proper setting is the smaller value.

This number, which can be considered the appropriate number of pulses under ideal conditions, is then weighted by multiplying by the fuzzy set evaluation that has been saved. This is the number of pulses that is commanded to the system for jet firings in the required direction. However, no more or less firings than is physically possible for the system are executed. Any additional ones are simply dropped since the evaluation procedure on the next cycle will command additional firings if they are still necessary.

To illustrate the number of pulses computation consider figure 3.

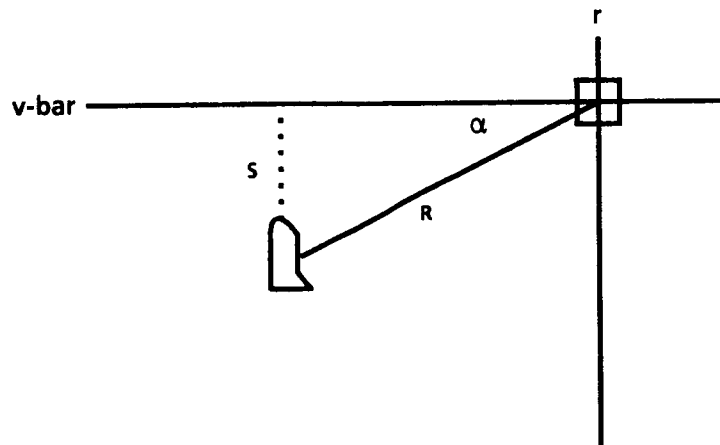


figure 3

In this case the shuttle is "low" with respect to the desired position on the \bar{v} bar. By using the shuttle-target range and angle α an approximate change in velocity can be computed using the equation

$$\Delta V = f(\alpha) \omega Rk - R\alpha$$

which relates range R in feet to required change in velocity ΔV in feet per second to move the shuttle up to the \bar{v} -bar, ω is the orbital rate, and k is a constant of proportionality. This estimate of ΔV is adjusted according to whether the shuttle is currently moving up or down relative to the target. The function $f(\alpha)$ is the fuzzy function corresponding to target "high" in the field of view and the number of pulses to be applied is given by

$$N = (\Delta V/d) * f(\alpha)$$

Here $\omega Rk/d$ represents the action function A referred to earlier and d is the current DAP setting. In a similar way fuzzy sets are used for controlling closing rates, out of plane angles, and elevation and azimuth rates.

These studies indicate a general approach to the automated rendezvous problem. In fact, they have implications in a general problem of vehicle control. In a problem of this type, "n" control rules would be modeled with fuzzy sets. Each of the fuzzy sets would be evaluated separately, the most critical identified, and an appropriate action determined. The decision making process would integrate all of the existing and new technology in the areas of expert system development tools and engineering control systems with the new fuzzy control methods.

RESULTS

Many different scenarios have been run with the automated system and performance with respect to flight profile and ΔV requirements have been very good. Not all automated scenarios were tested against manual control but those that were have performed better. For example, comparisons of ΔV requirements for a man-in-the-loop versus automated controller gave the results in the following table:

SCENARIOS	MAN-IN-THE-LOOP ΔV REQUIRED	AUTOMATED CONTROLLER ΔV REQUIRED
Stationkeeping at 150' for 30 minutes	0.54 ft/sec	0.1 ft/sec
V-bar approach from 500' to 40' 25 minute time interval	2.99 ft/sec	2.12 ft/sec

Noise free data was used for both cases since the intent is to simulate filtered and smoothed data.

To further demonstrate the capability of the system for proximity operation, all of the phases shown in figure 4 have been run (i.e., terminal phase rendezvous, labeled (1), V-bar approach, labeled (2), and separation, labeled (3)).

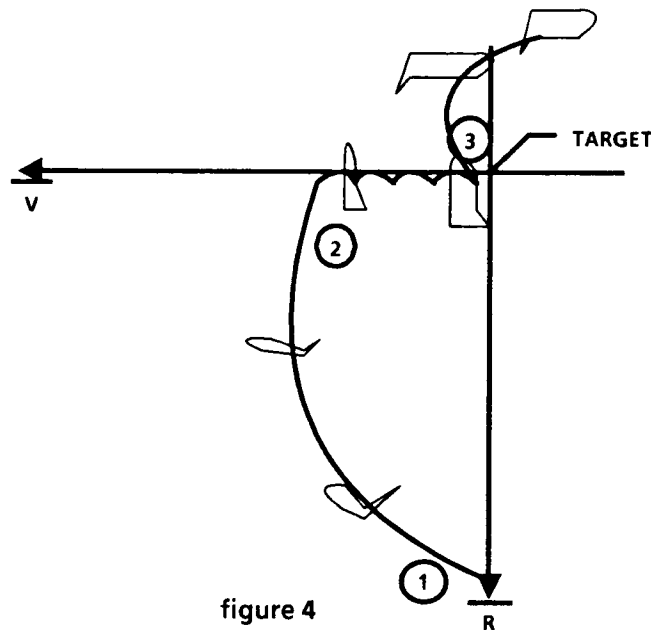


figure 4

The system will allow transition from one point to another in the proximity operations region. For example, one can transition from a point on the V-bar to a point on the r-bar and stationkeep, or one can transition from a point on the r-bar to a point on a general approach vector to the target that may or may not be in the plane of target motion.

STATUS AND CONCLUSIONS

The preliminary results of an automatic controller for a rendezvous vehicle proximity operations simulator that controls maneuvers based on fuzzy decision functions indicate the goal of complete autonomy is achievable. Indeed the results of tests of the controller have shown it is possible to simulate the common sense reasoning of a pilot using fuzzy decision functions to express rules obtained from experienced pilots and integrate this with more sophisticated engineering control concepts in such a way that an efficient system is achieved.

Many general proximity operations scenarios for rendezvous vehicle control have been run to test the system. In particular test runs have been made with the active vehicle both in and out of the plane of the target vehicle and above or below the desired approach path to the target. Approach angles to the target have been varied to show that approaches are possible along any vector to the target. Stationkeeping can be performed at any range or time and transition from one stationkeeping position to another is possible while keeping the relative range rate nulled. For example the system can support a maneuver from stationkeeping on the \bar{v} to stationkeeping on the \bar{r} .

As the approach is extended to other applications, or possibly to speed up use of the present application, it is realized that a fuzzy function chip of the type described by Togai [4] or Yamakawa [5] could be used to offload a great deal of the computation. This will be especially appropriate as the system is expanded to include larger and larger parts of the guidance, navigation, and control functions. It is intended to investigate the usefulness of such hardware as soon as it is available.

The current model of the controller assumes that the data it receives from sensors is smooth. It does not require extremely accurate data however. If sensors giving relative position and rates are very noisy or have large biases some type of filtering will be desirable but it will not have to be an extremely sophisticated filter.

The control system will be applicable to performance testing in a variety of rendezvous profiles and to determining the accuracy required for rendezvous sensors, as well as required redundancy in the system, and propellant requirements.

ACKNOWLEDGEMENT

The author would like to acknowledge Edgar Lineberry of NASA/JSC/MPAD for valuable suggestions of test scenarios, improvements, and generalization for this simulator. Acknowledgements are also due to Eric Von Mitchell who supplied the manual test runs for comparison and Sam Wilson and John Whynott who provided assistance in working out problems within the simulator where the autonomous controller resides.

REFERENCES

- [1] Lea, R. N., A Fuzzy Set Approach to a Navigation Decision Making Problem, Proc. of the 1985 Conf. on Applied Analysis, Univ. of Houston/Clear Lake, Houston, Texas, November, 1985.
- [2] Lea, R. N., Goodwin, M. A., and Mitchell, E. V., Automated Control Procedures for Shuttle Rendezvous Proximity Operations, Space Operations Automation and Robotics Conference, NASA/Johnson Space Center, Houston, Texas, August, 1987.
- [3] Zadeh, L. A., Fu, K. S., Tanaka, K., and Shimura, M., (eds.), Fuzzy Sets and Their Applications to Cognitive and Decision Processes, Academic Press, New York-London, 1975.
- [4] Togai, M., and Watanabe, H., Expert System on a Chip: An Engine for Real-Time Approximate Reasoning, IEEE Expert, Fall 1986.
- [5] Yamakawa, T. and Miki, T., The Current Mode Fuzzy Logic Integrated Circuits Fabricated by the Standard CMOS Process, IEEE Trans. on Computers, vol. C-35, no. 2, February, 1986.

AUTOMATED SATELLITE CONTROL IN ADA

Allan Jaworski and J.T. Thompson
 Ford Aerospace Corporation
 7401 D Forbes Boulevard
 Seabrook, Maryland 20706

Abstract

This paper describes the Advanced Ground Segment, a prototype satellite/payload operations control center workstation, which represents an evolutionary effort to improve the automation of control centers while improving software practices and supporting distributed control center functions. Multiple levels of automation are supported through a rule-based control strategy. The architecture provides the necessary interfaces and modularity for future inclusion of more sophisticated control strategies.

Introduction

A significant portion of a spaceflight mission's life cycle cost is associated with the development, maintenance, and operation of the ground control system. Moreover, as the life of a spacecraft increases, so does the percentage of ground cost to total project cost. The spacecraft complexity made possible by modern launch systems and flight technology has resulted in an increased operational burden and an increased risk of loss of spacecraft function through human error. These issues must be addressed as part of the preparation for the Space Station era. The preliminary Space Station Operation requirements clearly state:

"Flight and ground systems design shall consider automation for effective resource utilization ... Subsystems shall be automated to the fullest extent practical, using man's capability to provide a cost-effective alternative."

A modern payload/spacecraft operations control center can be quite expensive to build and operate. For example, the NASA Space Telescope Operations Control Center will involve a network of six large DEC VAX computers, seventeen VAX workstations, and multiple communications processors linked together by several networks. Over 1.2 million lines of custom software have been developed for the system. Ground operations will involve at least 39 full-time staff over the 20 year life of the Space Telescope.

Control requirements for the Space Telescope are exceedingly complex, as a result of the extensive redundancy and cross-strapping of the spacecraft. The state space of the flight system consists of a large number of discrete and analog parameters governed by complex interactions and control delays. For example, the DF-224, one of five types of elements in the Space Telescope Data Management System (DMS), has 3076 possible configurations. The Data Management Unit, another element of the DMS, has over a thousand possible configurations. When taken together these two units alone have over 3 million possible discrete configurations. The thermal model for the Optical Telescope Assembly integrates 240 analog sensor measurements, related in a time delayed manner to the DMS states (e.g., response of a thermal sensor to a heater state). When taken together, even this relatively small subset of spacecraft variables results in an overwhelmingly complex control problem. However, real-time tracking of these complex configuration states -- confirmation of commands, detection of anomalous conditions, and cross-verification of subsystem states -- without automation will be increasingly labor-intensive and error prone.

Complex Space Station era spacecraft such as the polar platforms being planned for earth sensing applications will present even more challenging control problems. Also, as spacecraft users demand more direct and transparent access to payload resources control center implementers will need to rely more and more on automation techniques to improve system response.

Significant savings in development, maintenance, and operations costs for these complex systems are, however, feasible. For example, estimates made in the initial phases of the project being reported here indicated that at least 60% of the long-term personnel budget for the Space Telescope Operations Control Center could be saved through the introduction of relatively simple automation techniques.

Software which represents nearly 70% of initial systems cost in ground control facilities and a larger proportion of ongoing systems maintenance is also an obvious target for improvement. Automated software systems clearly have a potential to reduce staffing requirements but any effort to develop such systems must weigh the benefits of automation against the potential costs of software development and maintenance. The Ada programming language and related methodologies and software development environments being developed by the Department of Defense are a promising means to controlling these costs.

This paper describes a portable distributed workstation architecture which uses Ada and artificial intelligence techniques to address these issues. It also describes a prototype which implements this architecture using an object-oriented design.

Objectives

Ford Aerospace has initiated a project to develop an advanced architecture and reusable software components which will support NASA's needs for highly reconfigurable payload and spacecraft control centers during the 1990's. Objectives of the Advanced Ground Segment (AGS) project are to develop an architecture which:

- o reduces the need for large numbers of spacecraft operators and schedulers;
- o minimizes danger of spacecraft or payload damage due to operator error;
- o supports distributed planning and scheduling of spacecraft/payload resources;
- o supports rapid but controlled access to payload services by operators and users;
- o is sufficiently modular to incorporate new automation techniques as they become available including classic control algorithms, rule and frame based expert systems, model-based reasoning, and neural networks;
- o is applicable to both small single workstation payload-oriented control centers and large-scale multi-workstation distributed control centers, and;
- o supports eventual migration of function to onboard processors.

System Architecture

In this section we describe the architecture of the currently implemented prototype which runs on MicroVAX based workstations and uses the DEC VAX Ada environment for software implementation.

Figure 1 is a pictorial representation of the AGS architecture as currently implemented in the prototype system. The method for representing the design is based on a widely used notation for Ada program designs developed by R.J.A. Buhr of Carleton University. Trapezoids indicate computer processes which may occur in parallel using the Ada tasking paradigm. Boxes with a clipped upper right corner represent Ada data types, each capable of replication and dynamic allocation to processors.

Separate collections of Ada tasks are used to provide emulation of a spacecraft and communications links (including line noise and dropouts). All telemetry and commands are generated and stored in a manner consistent with the most recent versions of the Space Station standards currently proposed by the Consultative Committee for Space Data Systems (CCSDS). The use

FIGURE 1. ADVANCED GROUND SEGMENT DESIGN

of these standards alone effectively automates one major task of spacecraft control centers, reconfiguration associated with changing communications formats.

Centralized functions are responsible for modeling and predicting spacecraft state, managing a data base of telemetry information, comparing expected to measured states, triggering automated functions based on measurements, and scheduling command loads. Plans are underway to provide additional capabilities for distributing the scheduling function based on a network-oriented scheduling protocol being developed by Ford Aerospace jointly with the University of Colorado at Boulder.

Each user or operator who logs onto the system is provided with a separate expert assistant responsible for checking of user privileges, initial checking of command sequences prior to a centralized command verification function, automated monitoring of specific functions, and generation of command sequences for submission to the command scheduler. Interfaces with the user subsystem have been engineered to allow future substitution of custom assistants. The current assistant strategy uses a rule-based expert system which is downloaded through a network from an offline Lisp-based system described below. Network interfaces for the user subsystems are engineered to allow distribution of functions over both local and wide area networks.

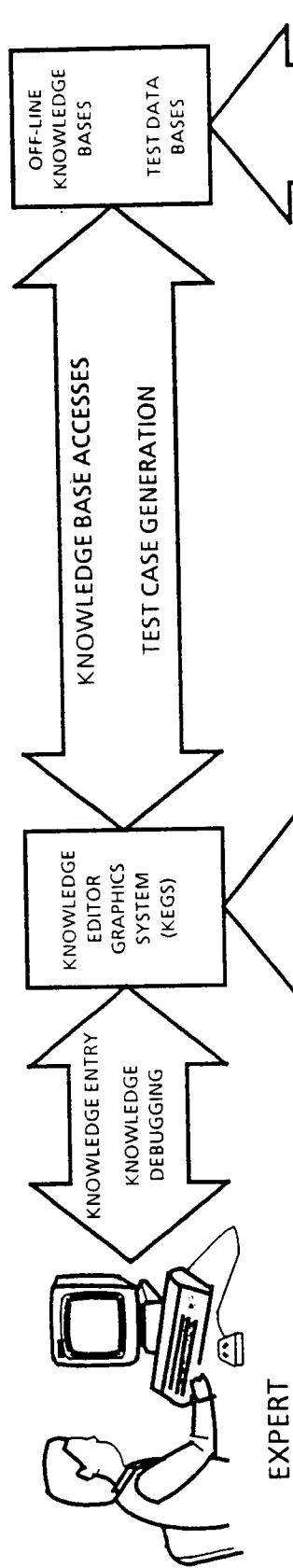
All system functions are configured through data bases which may be modified by suitably privileged users in real-time. Even graphics displays are configured through a simple data base addition or modification. This is a natural evolutionary improvement over current control centers which typically allow real-time reconfiguration of alphanumeric displays but rely on relatively static graphics. Efforts are underway to further improve reconfigurability by providing a MacIntosh-like resource editor which will allow users to rapidly construct custom displays out of reusable Ada components through a click-and-drag mouse interface.

Expert System Implementation

The expert system contained in the present system was created with the aid of the Ford Lisp-Ada Connection (FLAC), an integrated development environment designed to support direct entry and testing of rule-based knowledge on a Lisp machine and network downloading of a data base to an inference engine which has been implemented in the Ada language. The overall structure of FLAC is shown in Figure 2. FLAC consists of two components, the Knowledge Editor Graphics System (KEGS) and the Ford Ada Inference Engine (FAIE).

Knowledge is entered through KEGS, an easily learned knowledge base Computer Assisted Design (CAD) system which provides integrated features for rule development and knowledge base testing. An expert can use a set of menu- and mouse-driven

OFF-LINE SYSTEM



NETWORK
TERMINAL
SERVICES

ONLINE SYSTEM

NETWORK
FILE
TRANSFER

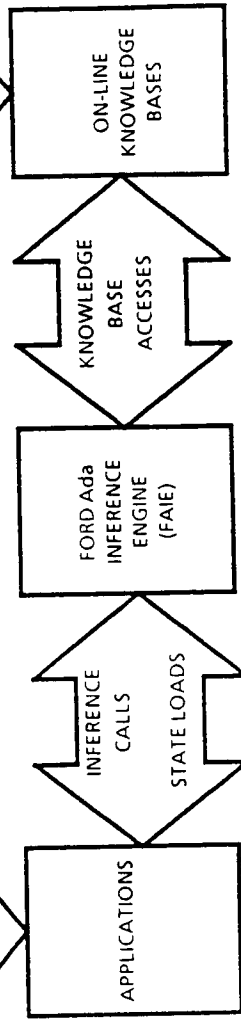


FIGURE 2. LISP ADA CONNECTION

resources to develop a knowledge base which is graphically represented as an and-or gate diagram. Tools are provided for the expert to rapidly enter, test, and debug knowledge base logic paths. The user interface is similar to those found in CAD systems for integrated circuit logic design.

The knowledge base can be downloaded to FAIE, an extremely fast portable Ada-based inference engine which is capable of firing more than 1500 rules per second on a MicroVAX II workstation. FAIE supports both forward and backward chaining modes of inference. The FAIE run-time environment has previously been used in a prototype of the Space Station Operations Management System implemented by McDonnell-Douglas.

Within the specific application domain of a workstation-oriented control center FAIE provides more than adequate performance for real-time monitoring and control of individual spacecraft subsystems. Our general approach to automation has been to focus on construction of multiple relatively simple expert systems to perform subsystem monitoring and avoid the construction of large multi-thousand rule expert systems which are difficult to test or run in real-time.

Substantial consideration was given to the level of autonomous control to be provided. Although some expert systems are provably better in judgement than human operators it is difficult to transfer trust to a computer system for judgements which might affect the health and safety of a spacecraft which may cost many hundreds of millions of dollars. This problem was attacked in two ways:

1. We limited the scope and size of the expert systems to relatively small rule bases which could be thoroughly tested for a large number of logic paths. Offline tools for this testing are provided within KEGS.
2. We support four basic levels of automation:
 - a. fully autonomous (no operator involvement)
 - b. automatic (operator is advised of command actions)
 - c. advisory (operator is advised of command actions and can edit commands before they are uplinked or appended to the command schedule)
 - d. fully manual (expert system is disabled)
3. A central command verification function is provided. Although this function is not automated in the current prototype we expect to eventually replace this function with a separate automated system.

The current systems provides adequate performance and characteristics to automate a large amount of the routine monitoring and commanding requirements for a control center. However, a significant amount of work in improving Ada-based expert system technology must be done before such a system can be trusted to perform non-routine satellite control or recovery

activities. Until such capabilities are available logic paths for many anomalous conditions must either result in an operator message or a recommended safe-hold strategy.

Future Plans

Future plans for the AGS system include:

- o Improving the distribution of its data base and control mechanisms -- The end objective is a fully distributed hierarchical planning and command architecture. This will be well-supported by the system's object-oriented design.
- o Optimizing the performance of the current software design -- Surprisingly, the Ada design performs as well as similar Fortran-oriented software in spite of its object-oriented layering for portability and reuse. However, substantial improvements in performance are feasible through the use of a number of design enhancements associated with the use of the Ada tasking constructs.
- o Adding additional artificial intelligence technologies -- Two separate efforts are planned, both with the aim of decreasing the software cost of automation:
 - A model-based reasoning system based on Ford Aerospace's Paragon tool for knowledge data capture and structural reasoning. This tool is currently being ported from its Lisp implementation to an Ada implementation.
 - A sentinel system based on a neural network emulation in Ada. The objective is to develop a robust assistant which can learn about anomalous telemetry conditions from previous examples and react to similar conditions.

Neural networks in many ways are the ideal control structure for automated control centers since they can be trained through a sample connection of anomaly scenarios and react to conditions which are similar but do not precisely resemble the training scenarios. However, it is difficult to fully test a neural network and visibility into the internal decision structure is somewhat limited. As a result we plan to focus primarily on the use of neural networks for monitoring telemetry streams and flagging conditions for other types of analysis such as model-based reasoning.

Conclusion

The AGS architecture is envisioned as the forerunner of future control centers to be built to Ford Aerospace. The long-term vision is to construct a control architecture whose components can eventually be freely replicated and distributed across geographic locations and across space and ground systems. The AGS effort has already had a strong influence on the design philosophies for Ford Aerospace's most recent control center designs for the Space Telescope and the GOES I-M satellite series.

Planning and Scheduling

Contingency Rescheduling of Spacecraft Operations

**Knowledge Based Tools For Hubble Space Telescope
Planning And Scheduling: Constraints And Strategies**

**The Proposal Entry Processor: Telescience Applications
For Hubble Space Telescope Science Operations**

**Candidate Functions For Advanced Technology
Implementation In The Columbus Mission Planning
Environment**

**A Rule-Based Systems Approach To Spacecraft
Communications Configuration Optimization**

**Integrated Resource Scheduling In A Distributed
Scheduling Environment**

CONTINGENCY RESCHEDULING OF SPACECRAFT OPERATIONS

Daniel L. Britt
 Martin Marietta Information and Communication Systems
 MS 4443
 P. O. Box 179
 Denver, CO 80201

Amy L. Geoffroy
 Martin Marietta Information and Communication Systems

John R. Gohring
 Martin Marietta Data Systems

ABSTRACT

Spacecraft activity scheduling has been a focus of attention in artificial intelligence recently. Several scheduling systems have been devised which more-or-less successfully address various aspects of the activity scheduling problem, though most of these are not yet mature, with the notable exception of NASA's ESP. Few current scheduling systems, however, make any attempt to deal fully with the problem of modifying a schedule in near-real-time in the event of contingencies which may arise during schedule execution. These contingencies can include resources becoming unavailable unpredictably, a change in spacecraft conditions or environment, or the need to perform an activity not scheduled. In these cases it becomes necessary to repair an existing schedule, disrupting ongoing operations as little as possible. Normal scheduling is just a part of that which must be accomplished during contingency rescheduling.

A prototype system named MAESTRO has been developed for spacecraft activity scheduling. This paper briefly describes MAESTRO, with a focus on recent work in the area of real-time contingency handling. Included is a discussion of some of the complexities of the scheduling problem and how they affect contingency rescheduling, such as temporal constraints between activities, activities which may be interrupted and continued in any of several ways, and different ways to choose a resource complement which will allow continuation of an activity. Various heuristics used in MAESTRO for contingency rescheduling will be discussed, as will operational concerns such as interaction of the scheduler with spacecraft subsystems controllers.

I. Introduction

Spacecraft in operation have a multitude of ongoing simultaneous activities, each having its own set of timing requirements, resource consumptions, and environmental conditions requirements and effects. When all operations are running as projected, and conditions and resource availabilities occur as predicted, spacecraft will operate smoothly. Frequently, however, some of these projections and predictions turn out to be wrong; some experiment may take longer to perform and more data transmission time than initially planned for, there may be a partial power loss, or some target of opportunity may arise, causing the spacecraft operators to attempt to add some activity to the timeline. All of these deviations from the set of operations originally scheduled can require revision of the initial operational schedule. These revisions must be done well and in a timely manner to ensure that the spacecraft continues to operate efficiently and safely.

In the following sections we describe some important aspects of the problem of revising a schedule in a contingency situation, and the solutions embodied in the prototype scheduling system MAESTRO. We begin with a description of the scheduling problem in general, then describe MAESTRO, focussing on its contingency handling mechanisms. The next section describes some of the problems associated with embedding a scheduler into an operational context, and we conclude with an indication of problems yet to be addressed.

II. Scheduling

Scheduling may be defined as the placement of performances of activities on a timeline such that those activities may actually be performed as placed. Activities include experiments, maintenance or repair of equipment, meal preparation and consumption, etc. In order to ensure that an activity can be performed as specified on the schedule, it must be verified that all constraints which are absolute requirements for the performance of that activity are met. These constraints may include resources and conditions requirements, allowable time windows, number of performances required, and timing relative to the performance of other activities. Other constraints, such as preferences in resource use or performance placement, may be considered as well.

Take as a simplified example the scheduling of a few hypothetical Space Station experiments - say one using a targeting

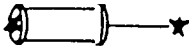



<div>TARGETING EXPERIMENT</div> <div></div>	<div>Resource requirements:</div> <table><tr><td>Power</td><td>Pointing</td><td>Calibration</td><td>Data Collect</td><td>Shut Down</td></tr><tr><td>Heat Rej</td><td>50</td><td>150</td><td>190</td><td>50</td></tr><tr><td></td><td>50</td><td>150</td><td>190</td><td>50</td></tr></table> <div>Conditions:</div> <div>Target must be available for data collection phase</div> <div>Vibration must be < x for calibration and data collection</div> <div>Atmospheric pollution must be < y</div> <div>Side effects: Creates vibration when rotating in pointing phase</div> <div>Time windows: Mon. - Fri. 11am - 6 pm (when ground support is available)</div>	Power	Pointing	Calibration	Data Collect	Shut Down	Heat Rej	50	150	190	50		50	150	190	50									
Power	Pointing	Calibration	Data Collect	Shut Down																					
Heat Rej	50	150	190	50																					
	50	150	190	50																					
<div>SPIDER EXPERIMENT</div> <div></div>	<div>Resource requirements:</div> <div>Continuous power & heat rejection, 25w each</div> <div>1 Crewmember (PO or PS) for observation phases</div> <div>Conditions:</div> <div>Observation phases vibration < z</div> <div>Observations must occur at least 10 minutes after "sunrise", during "daylight" only</div> <div>Coordination:</div> <div>Observation phase of this experiment should co-occur with the filming phase of the Public Relations filming</div>																								
<div>CRYSTAL SAMPLE GROWTH EXPERIMENT</div> <div></div>	<div>Resource requirements:</div> <table><tr><td></td><td>Set-up</td><td>Heat</td><td>Grow</td><td>Centrifuge</td><td>Analysis</td></tr><tr><td>Crew</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>Power</td><td>0</td><td>500</td><td>250</td><td>110</td><td>25</td></tr><tr><td>Heat Rej</td><td>0</td><td>200</td><td>500</td><td>110</td><td>25</td></tr></table> <div>Conditions:</div> <div>Sample growth < p vibration</div> <div>Sample analysis < q vibration</div> <div>Gasses and liquids: 20l Helium</div> <div>5l Argon</div> <div>2l H₂O</div> <div>Side effects: Centrifuging induces X vibration</div>		Set-up	Heat	Grow	Centrifuge	Analysis	Crew	1	0	0	1	1	Power	0	500	250	110	25	Heat Rej	0	200	500	110	25
	Set-up	Heat	Grow	Centrifuge	Analysis																				
Crew	1	0	0	1	1																				
Power	0	500	250	110	25																				
Heat Rej	0	200	500	110	25																				
<div>PUBLIC RELATIONS FILMING</div> <div></div>	<div>Resource requirements:</div> <table><tr><td></td><td>Set-up</td><td>Film</td><td>Break-down</td></tr><tr><td>Crew</td><td>1</td><td>1</td><td>1</td></tr><tr><td>Video Cam</td><td>1</td><td>1</td><td>1</td></tr><tr><td>Power</td><td>0</td><td>125</td><td>0</td></tr></table> <div>Conditions:</div> <div>No venting during filming</div> <div>Filming < x vibration</div> <div>Coordination:</div> <div>Filming phase must co-occur with spider observations</div> <div>1 roll film</div>		Set-up	Film	Break-down	Crew	1	1	1	Video Cam	1	1	1	Power	0	125	0								
	Set-up	Film	Break-down																						
Crew	1	1	1																						
Video Cam	1	1	1																						
Power	0	125	0																						

Figure 1. Example experiments for the Space Station, with their resource, conditions, and coordination requirements.

instrument, another involving a crew member's observation of some spiders, and another involving the generation and centrifuging of some samples (see Fig. 1). For each of these activities certain resource requirements must be met in terms of power (for the targeting and sample experiments), crew time (for the spider and sample experiments), etc. Insofar as these resources are limited, different activities may compete for these resources, requiring the coordination of the activities. Conditions for each of the experiments must also hold - the targeting instrument must be able to acquire its target and may require a minimum vibration, while the centrifuging phase of the sample experiment may generate a certain amount of vibration. Additionally, there may be a requirement for filming of a crew member performing observations in the spider experiment for a publicity film, and this will

require coordination of the film's timing, resource, and conditions requirements with those of the spider experiment. The process of producing a schedule which provides the necessary coordination of resources and conditions for these activities can be quite complex.

In addition to assuring that activities may be performed as placed on a timeline, scheduling is involved with the attempt to produce a "good" schedule, getting as many activities performed as possible within limits on resource availabilities, respecting differing priorities various activities may have associated with them, etc. This is a crucial function, especially with respect to scheduling of activities aboard spacecraft, as the cost of providing the opportunity to perform these activities is astronomical.

Scheduling includes not only initially generating a schedule to be followed, but also altering that schedule to reflect any changes in the assumptions upon which that schedule was based. Contingencies which can occur include equipment breakdowns, resource availability changes, a change in spacecraft conditions or environment, the need to perform an activity not scheduled, or simply an activity requiring more time or resources than anticipated. The scheduling problem in these cases becomes one of repairing an existing schedule, disrupting ongoing operations as little as possible. Contingency operations typically include all the complexities of initial scheduling with the additional need to perform heuristically-guided unscheduling and alteration of performances in progress at the time of the interruption.

To continue with the example above, suppose a schedule is being executed wherein the principal investigator for the targeting experiment finds an unexpected opportunity to gather valuable and seldom-accessible data. This opportunity may require repointing the instrument during a scheduled spider observation, causing vibration which would interfere with that observation. The scheduler must revise the timeline, rescheduling the observation around the pointing phase of the targeting experiment. If that observation was to be the subject of a public relations film, that filming also would need to be rescheduled. These reschedulings may free some resource(s) needed by the crystal growth experiment, which could then be added to the schedule.

III. The MAESTRO Scheduling System

Much of the general scheduling problem is addressed in the MAESTRO scheduling system, described below.

Activities.

During scheduling, the entities placed on the timeline are activities. Activities within MAESTRO are represented hierarchically. An activity group is a set of activities representing different ways to accomplish a particular goal. An activity, in turn, is a linear sequence of subtasks which, when performed in the order specified, satisfies that goal. A subtask is a portion of an activity whose resources and conditions requirements do not vary over its duration. That duration can vary, as can delays between subtasks. A subtask description includes its minimum and maximum duration, the minimum and maximum delay allowable from the end of the preceding subtask, the ways the subtask can be dealt with if interrupted, and a representation of the various constraints which must be satisfied in order to execute the subtask. The constraints MAESTRO handles are described below.

Constraints.

Constraints representable within MAESTRO on the performance of an activity are of four basic types:

- 1) The availability of resources or conditions necessary to the performance of a subtask. There are several kinds of constraints within this category. Rate-controlled resources are those whose availability returns the moment a subtask consuming them ends. Examples of this type are crew time, thermal rejection, electrical power and equipment. These can be contrasted with consumables, which, once depleted, stay depleted until some activity specifically replenishes them. Water, liquid nitrogen and lubricating fluids are examples of this type of constraining resource. Conditions, another kind of constraint, are states the spacecraft must maintain in order to perform a subtask, and include spacecraft attitude and position, temperature ranges, acceleration, vibration, etc. In general, conditions cannot be consumed by an activity requiring them, which differentiates them from rate-controlled resources.

Some of these constraints can be satisfied by more than one resource or condition. An example of this is the case where a subtask could be performed by either of two crew members trained to use a particular piece of equipment, but not by any of the

other crew members. This is referred to as a resource disjunction, a case where one resource or another can satisfy a requirement. The existence of a resource disjunction in a subtask description greatly increases the complexity of finding all times during which a subtask can run, as opportunities to perform the subtask depend on which resource is chosen. This can be further complicated by the fact that a resource choice in one subtask can control that in another, e.g. the crew member who performs the calibration of an instrument should be the same one who read the manual at the start of the activity.

2) Constraints relating the performance of two subtasks in the same activity. Sequencing of subtasks, their durations, and the delays between them are included in these, as are more general ways of relating subtasks in an activity. There may be a minimum separation between the first and fourth subtasks, for example, or a maximum duration for the whole activity.

3) Constraints relating the performance of a subtask in one activity to that of a subtask in another. These can be used to ensure that one subtask is performed before or after another, or that two subtasks start or end within some time period of one another. Multiple constraints of this type can be specified between two subtasks as long as they are not contradictory. For example, one subtask could be forced to begin and end at the same time as another.

4) Constraints relating the performance of an activity or subtask to some event or interval on the timeline. An interval may be specified during which an activity or a subtask must be performed, or one can specify a time interval during which a subtask must start or end.

Schedule generation.

MAESTRO creates a schedule by repeatedly executing three steps, referred to as the select-place-update cycle. The first step involves evaluating every activity requested for scheduling with respect to a set of selection criteria, and choosing one activity to schedule on that cycle. These criteria include the base priority associated with each activity, the percentage of performances requested that have been scheduled for each, and the relative constrainedness of each. Relative constrainedness is a rough measure of how many different opportunities each activity has to be placed on the schedule. These criteria are combined using user-selectable weightings which reflect the importance of each criterion to the user. An activity chosen for scheduling will have higher priority, a lower percentage of requested

performances scheduled, and/or fewer opportunities to be scheduled than other activities.

Once an activity has been chosen to be scheduled, one performance of it is placed on the schedule. The calculation resulting in the measure of constrainedness mentioned above actually determines all allowable start and end times for all subtasks in each activity. This information can be used during placement to position a performance according to soft constraints (preferences) imposed by a user. He can, for example, maximize a data collection subtask, or can schedule a activity as early or as late in the scheduling period as possible. If there is a resource disjunction in a subtask's requirements, a preference can be specified and adhered to. A set of possibly contradictory soft constraints can be specified, along with an ordering in their importance. In order to pay attention to a preference for a maximum data collection duration, for example, the scheduler may have to schedule an activity later than crew use preferences would dictate.

The final step in the scheduling cycle involves updating resource availability profiles to reflect the activity's consumption of resources. The cycle then repeats for as long as the user wishes or until there are no opportunities to schedule any activity. The combination of weightings on selection criteria and attention to soft constraints during placement allows the scheduler to be tuned for a variety of scenarios.

Under certain circumstances the scheduler will perform a fourth step in the cycle described above. The calculation which determines all allowable start and end times for each subtask, called opportunity calculation, recognises differences in priorities among activities. If opportunity for a high-priority activity cannot be found, the scheduler may choose to ignore the projected resource use for some lower-priority activities, placing the high-priority activity in such a way that resource overbookings occur. This necessitates exercising a portion of the contingency handling function described below in order to remove these constraint violations. This typically requires that one or more of the lower-priority activities be unscheduled. Once they are removed, the scheduler can proceed to the next iteration of the cycle.

Contingency operations.

The previous paragraphs described how MAESTRO creates or adds to a schedule. Additionally, there are a number of situations in which a schedule must be altered to accommodate

various changes. It may become known that resource or conditions availabilities will change or have changed, or that an activity not previously known about needs to be added to the schedule. These situations are handled within MAESTRO by a process similar to that used during initial schedule generation. The system repeatedly executes a cycle consisting of four steps: 1) detection and quantification of a resource over-use, or just detection in the case of a conditions constraint violation, 2) selection of an activity performance to perturb which will alleviate the problem to some extent, 3) unscheduling or alteration of that activity performance on the schedule, and 4) checking and possibly unscheduling all activity performances temporally constrained by the perturbed activity. This process continues until no resource or conditions constraint violations remain, and is described in more detail below.

The scheduler first determines the exact extent of the resource over-use, quantifying it so that each activity's use of the resource can be compared to the over-use.

Next the scheduler evaluates all activities using that resource during the time it is overbooked. Each activity is rated according to a set of criteria designed to determine what activities to alter or unschedule to solve the problem with the least impact on the schedule. These criteria include how well an activity's use of the resource fits the amount of overbooking, whether the activity is in progress or not, the activity's priority, amount of crew involvement, use of other resources, other opportunities to be scheduled, success level, and others.

The activity chosen to be perturbed is then either unscheduled, if it has not yet begun, or altered to reflect its progress and the necessary changes to it, if it is already executing. An activity which is unscheduled may be rescheduled after all constraint violations are removed.

There are several ways in which a performance of an activity on the timeline may be altered, and these must be specified by the user of the system when first describing the activities. An activity which is interrupted during the performance of a subtask may be continued in any of several ways. It may be possible to just continue that subtask when resources become available, and there may be a maximum delay before continuing. Alternatively, it may be possible to just skip the rest of that subtask, going on to the next. This might be the case with a data collection subtask wherein the minimum required amount of data has been collected. Finally, the interrupted

subtask may be restarted, proceeding from the end of the previous subtask.

Certain resources being used by a subtask affected by a contingency may be able to be replaced with others, allowing the subtask to continue uninterrupted with a different resource complement. In this case the alteration just involves changing subtask descriptions and resource availability profiles to reflect this change in resource use.

Whether any of these options - to continue, skip or restart the interrupted subtask, or to switch resources - is viable depends upon the nature of the interrupted subtask and those around it, and upon the specific point within the subtask at which it was interrupted. The user must specify the viable options for contingency alteration when describing the activity.

Finally, all activities constrained by the activity whose performance is altered or unscheduled must be checked to see if those constraints have been violated. For example, suppose a data collection subtask in one activity must follow a calibration subtask in another which gets altered such that it ends fifteen minutes later. The data collection subtask may have to be delayed. Since it is part of an activity, other subtasks in that activity may need to be moved. It could even happen that one of these constrains a subtask in a third activity. All such constraints must be checked, and alterations made to the schedule when they are found to be violated.

When it is determined that an activity not scheduled must be added to the timeline, the scheduler first tries to find a way to schedule it which will not disturb anything already scheduled. If no opportunity exists, MAESTRO will try to find opportunities which will result in only lower-priority activities being perturbed, and if found will unschedule or alter one or more of those.

The last thing MAESTRO tries to do after removing resource over-uses in a contingency is to try to schedule any activities whose requests have not been fully met, possibly using resources released when some other activity was altered or unscheduled.

IV. Scheduler-subsystems interactions

There is a difficult aspect of handling contingencies aboard spacecraft which has not yet been mentioned. The scheduler creates a schedule but does not execute it or even monitor its

execution. To be useful, a schedule must be communicated to the entities responsible for carrying out the activities scheduled. In many cases these entities will be humans, either ground control personnel or crew members. In other instances various subsystems will automatically initiate and carry out actions according to the schedule. Any of these entities may also occasionally need to perform actions which are at odds with the schedule, and the effects of these decisions must be communicated back to the scheduler. It then can revise the schedule such that ongoing operations will be minimally perturbed. Thus the scheduler must be interfaced with a variety of other systems.

MAESTRO has been applied to a variety of related domains. In one application the scheduler is interfaced directly with a power management and distribution (PMAD) breadboard at Marshall Space Flight Center [1]. This breadboard is intended to simulate the electrical power system onboard a Space Station Module (SSM), and as such incorporates a high degree of automation [2]. Given a schedule and some additional information, the PMAD system extracts a list of power system events, times at which a power system component must change its state to support some activity onboard the SSM. This list is used to control switching at the lowest levels in the PMAD system. It also contains power level and current information which is used to detect faults in the system.

In the event of a fault, activities may be interrupted or their resource use altered. Activities may no longer be able to access the resources they were scheduled to use. Changes such as these can cause the schedule to become invalid. Activities may be begun which cannot be completed, or these changes may interfere with others already in progress. The scheduler is equipped to handle these problems, as described previously. However, there are a number of timing issues which must be recognized and dealt with, four of which are described here.

Reaction Time.

While the scheduler is making changes, the power system and other subsystems will be trying to continue execution of an old and possibly invalid schedule, which can result in a cascade of faults registered by these subsystems. The scheduler must become informed of faults and produce an altered schedule as quickly as possible to minimize the disruption to ongoing operations caused by the fault. Fast reaction time is thus an important aspect of efficient handling of contingencies.

Implementability of Revisions.

Not only must the scheduler react quickly, but it also must not make changes to the schedule which are not implementable. Suppose the scheduler is informed that a fault occurred, interrupting an activity at 10 o'clock, and generates a revised version of the schedule which calls for the activity to resume using a different resource complement 5 minutes later. However, it requires 8 minutes to finish revising the schedule and communicate it to the subsystems and personnel executing the activity to be resumed. At the point the schedule is received it is already invalid. In this case the scheduler must make a projection as to when the schedule could be acted upon and what states the various systems will be in at that time. Using this information it can revise the schedule to be consistent with the state of the spacecraft not at the time of the fault, but at the time the new schedule is in place.

Consistency of Revisions with Spacecraft State.

Another timing issue must also be dealt with for contingency operations to proceed smoothly. A fault within the power system may require some sort of testing by power system software or personnel, or may result in a temporary loss of power in a non-faulted branch of the power system which could be remedied fairly quickly. If an activity is interrupted by such a fault, the scheduler may attempt to revise the schedule using information about the power system gathered while it is in an unknown state or a state of flux. In this situation it is likely that the revised schedule will be in error, and may cause further problems for the power system. Thus it is important that a steady and known state be reached and communicated to the scheduler before it revises the schedule.

Communication of Variance from a Schedule.

Faults or other unscheduled events within a power system often require that power users (loads) be turned off immediately, even though the hardware supplying that power is in operating condition. This can happen when a power source outside the module reduces its output, or when a high-priority activity must be begun immediately and requires power some other activity is using. In these cases the scheduler does not have time to alter the schedule and communicate it to the power system. The PMAD system just opens some switches according to a dynamic prioritization scheme, interrupting activities using power from those switches. This is known as load shedding. When a situation arises requiring load shedding, the scheduler must be capable of determining which

activities were interrupted, and take this into account when fixing the schedule. There are various methods by which the scheduler can be apprised of the new situation, and perhaps the most efficient of these is a direct communication from the PMAD system to the scheduler. The scheduler can then determine what other subsystems are affected and communicate changes to them. In general, there is a large volume of information to be passed between the scheduler and the various subsystems, necessitating a fairly direct communications path between these systems.

V. Conclusion

This paper has addressed some of the more fundamental issues related to rescheduling of spacecraft operations in contingency situations. There is much work which still needs to be done to further the concepts introduced here. The prototype scheduling system named MAESTRO continues to be a good vehicle for exploring and finding solutions for problems involving scheduling and rescheduling, and we intend to continue expanding its capabilities. We deem it especially important to study the aspects of rescheduling involving interactions between the various systems and personnel responsible for providing information, making decisions and carrying out operations onboard spacecraft.

VI. References

- [1]. Britt, Daniel L., John R. Gohring & Amy L. Geoffroy. The Impact of the Utility Power System Concept On Spacecraft Activity Scheduling. Proceedings of the 23rd IECEC, 1988.
- [2]. Miller, William D. & Ellen Jones. Automated Power Management and Distribution Within a Space Station Module. Proceedings of the 23rd IECEC, 1988.

**Knowledge Based Tools for
Hubble Space Telescope Planning and Scheduling:
Constraints and Strategies**

Glenn Miller¹

Astronomy Programs, Computer Sciences Corporation

Mark Johnston, Shon Vick and Jeff Sponsler
Space Telescope Science Institute²

Kelly Lindenmayer¹

Astronomy Programs, Computer Sciences Corporation

3700 San Martin Dr.
Baltimore, MD 21218

Abstract

The Hubble Space Telescope (HST) presents an especially challenging scheduling problem since a year's observing program encompasses tens of thousands of exposures facing numerous coupled constraints. This paper discusses recent progress in the development of planning and scheduling tools which augment the existing HST ground system. General methods for representing activities, constraints and constraint satisfaction, and time segmentation have been implemented in a scheduling testbed. The testbed permits planners to evaluate optimal scheduling time intervals, calculate resource usage and to generate long and medium range plans. Graphical displays of activities, constraints and plans are an important feature of the system. High-level scheduling strategies using rule based and neural net approaches have been implemented.

¹ Staff member of the Space Telescope Science Institute

² Operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration

1. Introduction

NASA's Hubble Space Telescope (HST) will provide important new capabilities for astronomical observation. HST will orbit the Earth above the distorting effects of the atmosphere, allowing unprecedented angular resolution, sensitivity and wavelength coverage using six scientific instruments (for more details, refer to Hall 1982). The Space Telescope Science Institute (STScI) is responsible for conducting the science operations of the HST, including planning and scheduling observations. Planning and scheduling HST observations is a particularly challenging problem for several reasons: A year's observing program will consist of a large number of activities (about 30,000 exposures of approximately 3,000 celestial targets). Many constraints must be satisfied, including proposer specified constraints (e.g. timing, precedence), orbital viewing constraints (e.g. Earth, Sun and Moon occultations), and spacecraft power and communications constraints. Detailed schedules must be prepared days to weeks in advance in order to obtain communications contacts and to generate spacecraft computer command loads. Schedules must be repaired to account for disruptions due to unpredictable astronomical events (e.g. a supernova) and due to spacecraft anomalies.

In this paper we describe the Science Planning Interactive Knowledge Environment (Spike) System planning and scheduling tools which are being developed at the STScI to augment existing ground system scheduling capabilities. Initial work has resulted in the development of an "Exposure Evaluation Tools" testbed which provides both manual and automated tools for long term scheduling.

The next section provides a brief overview of the HST planning and scheduling problem. The reader should consult Miller, et al. (1987) for more details. Section 3 describes the problems of long term planning for HST. Section 4 focuses on Spike's Exposure Evaluation Tools and how they provide the necessary functions for long term scheduling. Automated strategies for scheduling are the topic of section 5. The development methodology, including the use of artificial intelligence techniques and rapid prototyping are discussed in section 6. The interfaces between the Spike system and other ground system components are described in section 7. The last section discusses some planned extensions of the current system and the application of these tools to other scheduling problems.

2. Overview of HST Planning and Scheduling

An astronomer wishing to observe with the HST submits a scientific observing proposal. Based on the the recommendations of a peer review committee, the Director of the STScI selects which proposals are awarded observing time and assigns each proposal to one of three priority categories: high, medium and supplemental. Unless precluded by unforeseen technical problems, all high and medium proposals will be executed. The difference between the high and medium categories is that medium priority observations may be rescheduled to accommodate rescheduling of a high priority observation. High and medium priority proposals will consume about 70% of the estimated observing time. The supplemental proposals form a pool used to fill out the remainder of the schedule and the choice of a particular supplemental proposal is likely to be based on scheduling and operational considerations. The supplemental pool oversubscribes the available time, so there is only a moderate chance that a particular supplemental program will actually be executed.

At this point, the scheduling process begins. A year's scheduling pool of about 300 proposals comprises tens of thousands of exposures on a few thousand targets. Proposal information is contained in the Proposal Entry Processor (PEP) System (Jackson, et al. 1988), which provides tools for entry, editing, evaluation, selection and transformation of proposals. A proposal includes target specifications (position, brightness, etc.) and a list of exposures (target, instrument, operating mode, exposure time, etc.). In order to express scientific constraints on the exposures, a

proposal can specify a wide range of properties and inter-relationships. For example, exposures may be designated as acquisition or calibration exposures. Some exposures must be executed at particular times or at specific spacecraft roll angles. Ordering and grouping of exposures may be specified as well, and these links may couple exposures separated by many weeks or months. Exposures requiring low background light conditions are identified for execution when HST is in the Earth's shadow.

In addition to the constraints imposed by the proposer's scientific program, there are a large number of other constraints which must be considered. Many orbital factors exert a strong influence on scheduling: targets are occulted (blocked) by the Earth for up to 40^m each 95^m orbit. Observations cannot be taken during HST's passage through the South Atlantic Anomaly (SAA), which may last 20^m. The telescope cannot point too closely to the Sun, Moon or bright Earth limb. The roll orientation of the spacecraft is constrained in order to maintain correct power and thermal balance. Communications with HST is via the Tracking and Data Relay Satellites (TDRS) and links will be available for only part of an orbit (this also limits the amount of real time interactions with the HST and instruments). Slews are relatively slow (90° in ~15^m), so efficient ordering of telescope pointing is important. Available electrical power limits the number of instruments that can be in "standby" or "operate" modes, and cycling between instrument can take several hours. (Refer to Miller, et al. 1987 for details.)

As a consequence of these and other factors, the operation of HST is almost entirely pre-planned. Long range plans must ensure the overall feasibility of the program. Short term plans must be consistent with the long range plan. Changes to the schedules caused by unexpected astronomical events (e.g. a supernova), instrument anomalies, changes in TDRS schedules, etc. must be accommodated and factored into the long term plan and to related exposures.

Currently, HST planning and scheduling is supported by the Science Operations Ground System (SOGS) Science Planning and Scheduling System (SPSS), which was developed by TRW. Initial population of the SPSS scheduling data structures is via the PEP Transformation Subsystem, an expert system which takes the astronomer-oriented proposal from the PEP system and creates the detailed implementation parameters required by SPSS (Rosenthal, et al. 1986). While SPSS has been successfully used to generate detailed schedules of a few days duration, there are several factors that severely limit its use on the long-range planning problem: SPSS scheduling algorithms only examine a few possible times to schedule exposures, and can therefore easily miss good scheduling opportunities. SPSS always considers detailed orbital events and conditions, even when they are uncertain or unpredictable. This makes it computationally very expensive to construct and evaluate long-range plans. A significant number of scheduling constraints are not considered by SPSS, and, because of the design and implementation of the system, they are difficult to add to the software. Coordination of related exposures which fall into different short term plans is essentially a labor intensive, manual process. Even for short term plans, the throughput of the overall system (people plus software) remains a concern.

Work towards enhancing the scheduling capabilities of the HST ground system is directed along two lines: 1. increasing the performance, reliability, maintainability and functionality of the existing SPSS software, and 2. creating new tools which augment the existing software. The latter work, the Spike system, is the subject of this paper. These two efforts are being carefully coordinated. The current focus in the Spike system is the development of long term planning tools, the first phase of which are the Exposure Evaluation Tools.

3. Long Term Planning

For HST science operations there are several key considerations for long term planning:

- plan must cover a long time interval (multi-year)
- planning is far in advance of execution, and many constraints can not be predicted in detail in advance
- plan must incorporate a large number of exposures
- constraints can couple exposures separated by long time intervals
- replanning will be required

Multi-year planning is an essential part of HST science operations. The basic observing cycle is one year long, and it is necessary to consider observations in preceding and succeeding cycles: Priority observations from the preceding cycle may, for various reasons, not be executed and will be "carried over" into the current cycle. Although most proposals will be completed in one cycle, some proposals are for multi-year observation programs, and the effect of these on future observing cycles must be considered. Additionally, in the first few years of HST operations, the Science Verification (SV) proposals from the Instrument Teams are mixed with General Observer (GO) proposals from the scientific community. Long term planning will be vital to ensure that short term schedules are consistent with the overall SV and GO objectives. Long term planning is also necessary to evaluate the effects of changes in the spacecraft, e.g. replacement of scientific instruments, slow decay of electrical power, etc.

The position of HST within its orbit can be predicted accurately about 3 months in advance. Beyond this, the in-track error grows so large that timing of events which depend on HST position (e.g. occultation by the Earth, SAA entry and exit) cannot be accurately predicted. A primary cause of this is fluctuations in the atmospheric density at HST's altitude due to fluctuations in solar activity. Fortunately, the orientation of plane of HST's orbit can be forecast with reasonable accuracy about one year in advance. This allows an average treatment of certain constraints. It should be noted that due to the large number of exposures in a long range plan, even if it were possible to predict HST orbital events with infinite accuracy, it would not be desirable to do so. A simpler approach for some constraints reduces the amount of computation while supplying the needed accuracy for long term plans. We consider the long term planning problem to range from several years to approximately 3 months before execution of the exposure.

South Atlantic Anomaly (SAA) passage serves to illustrate how constraints dependent upon the HST position can be meaningfully treated in an average sense for long term planning. The exact times of SAA entry and exit depend on the orbital location of HST. However, the fraction of time per day a target is unocculted by the Earth while the HST is outside the SAA depends only on the orientation of the plane of the orbit, and therefore can be estimated in advance (Sherrill 1987). In other words, for a particular day in the future, we cannot know the absolute time of SAA-free observing opportunities, but the number and duration of such opportunities can be known and used in a long term plan. The effects of scattered light from the Earth, Sun and Moon on faint targets can be treated in a similar manner.

Another important category of constraints are those which constrain the long term plan itself. This includes constraints on the consumption of various limited resources such as time, data volume, power, TDRS contacts and real-time operations. The efficiency of a plan is also another important metric in construction a long term plan.

The notion of hierarchical planning from a coarse to a fine level of detail has been explored in many scheduling problems (see, for example, Smith, Fox and Ow 1986) and will be useful for HST scheduling. Long term plans will span a few years, with perhaps a two month time resolution (two months is the precession period of the HST orbital pole, so this interval will encompass a complete

cycle of orbit dependent scheduling opportunities such as SAA passage and continuous viewing). As the long term plan is populated, subplans will be scheduled with perhaps a week's time resolution. Sufficiently detailed plans will then be sent to SPSS for detailed scheduling.

Replanning of two distinct types will be necessary. In going to a more detailed level (say a two month plan to a one week plan), it may be found that the high level plan was overly optimistic and that some observations allocated to a particular week cannot be executed due to some constraint. In this case, information from the lower level plan must be used to modify the higher level plans and the effects propagated (perhaps to other two month plans due to linkages between observations). The second type of replanning occurs when an observation fails to execute properly. For example, one of the stars in a guide star pair may be a binary star, which prevents tracking by the HST. Not only will it be necessary to replan that observation, it will be necessary to examine the effects on other observations which attempt to use that guide star. When an instrument shows some unexplained behavior, it will be necessary to suspend normal observations, schedule the necessary diagnostic operations and replan the suspended observations for some point in the future. Observations of targets of opportunities (e.g. a comet or supernova) will also cause schedule disruption and replanning.

Given this view of long term scheduling for the HST, the next section presents the Spike Exposure Evaluation Tools, demonstrating how they provide the required capabilities.

4. Spike Exposure Evaluation Tools

The Exposure Evaluation Tools were designed with the following features:

- uniform representation and manipulation of scheduling constraints
- express human value judgments and tradeoffs ("shades of gray" are handled as well as go/no-go criteria)
- easy to modify relative importance of constraints, and to add new constraints
- the interaction of constraints and tradeoff options are visible to human planners
- provide a means to track resource usage
- effective user interface
- ability to build automated scheduling tools on top of exposure evaluation tools

4.1 Activities, Clusters, Constraints and Suitabilities

An **activity** is the lowest level scheduling entity and is used to represent exposures (or possibly groups of exposures) and other planned actions such as instrument configurations, slews, communications contacts, etc. Exposures have a very few properties: duration, constraint list, priority and a flag to mark if the exposure is executed. As the duration of an activity may depend on its time of execution and relationship to other activities (e.g. slew time, exposure time, instrument configuration), the duration of an exposure is implemented as a function, not a constant.

Activities are grouped into **scheduling clusters**. These are the lowest level entities which can be scheduled. Scheduling clusters can represent SPSS scheduling units, branching sequences, logical components of an observing proposal, etc. To preserve scheduling flexibility, ordering of activities within scheduling clusters is not required. The default is one activity per scheduling cluster. Clustering reduces the number of entities to be considered by the scheduler.

A **constraint** is any factor that effects when it is possible or desirable to plan activities. This includes such **strict** constraints as "never point the HST closer than 40° to the Sun" and **preference** constraints such as "its preferable to execute the observation when 3 independent pairs of guide stars are available, but one pair is acceptable". There are two categories of

constraints: activity and segment constraints. **Activity constraints** limit the opportunities for a particular activity. An **absolute activity constraint** is independent of when any other activities are planned (e.g. Sun avoidance, orbital dark time, guide star availability, roll). A **relative activity constraint** relates two or more activities and depends on the suitabilities of other activities (e.g. precedence constraints, minimum and maximum time separation between exposures). Absolute activity constraints are fixed when planning begins, while relative constraints change as activities are fixed in the scheduling process. Segment constraints represent limitations on the overall plan and are defined later.

A fundamental component of any scheduling system is the representation of constraints. Constraints are represented in the Spike system as **suitability functions** (see Figure 1). A suitability function gives the desirability of starting an activity at a particular time. A suitability of zero means that a start at that time is forbidden, while a positive suitability indicates that the activity can begin at that time. A suitability of 1 is defined as the nominal suitability, with suitabilities greater than 1 indicating a more favorable starting time and suitabilities less than 1 indicating a less favorable starting time. A suitability function can be considered as a generalization of binary planning windows. The fact that suitabilities are not limited to a binary "yes/no" allows a powerful and natural way to express preferences and allowable tolerances on constraints. A suitability can be interpreted as the degree of constraint satisfaction. The formal properties of suitability functions are discussed in more detail in Johnston (1988b).

In the Spike software, constraint suitabilities are implemented as non-negative piecewise-constant functions (PCFs). This class of functions was chosen since it allows a particularly efficient representation of constraints and propagation of the effects of constraints. PCFs are closed under the operations of addition and multiplication, i.e. the sum or product of two PCFs is another PCF.

An important feature of the Spike software is that new constraints can be readily added to the system, or invoked as planning proceeds from long range to medium range to short range planning. Human judgment is required to establish the shapes of the suitability functions and the scale factors.

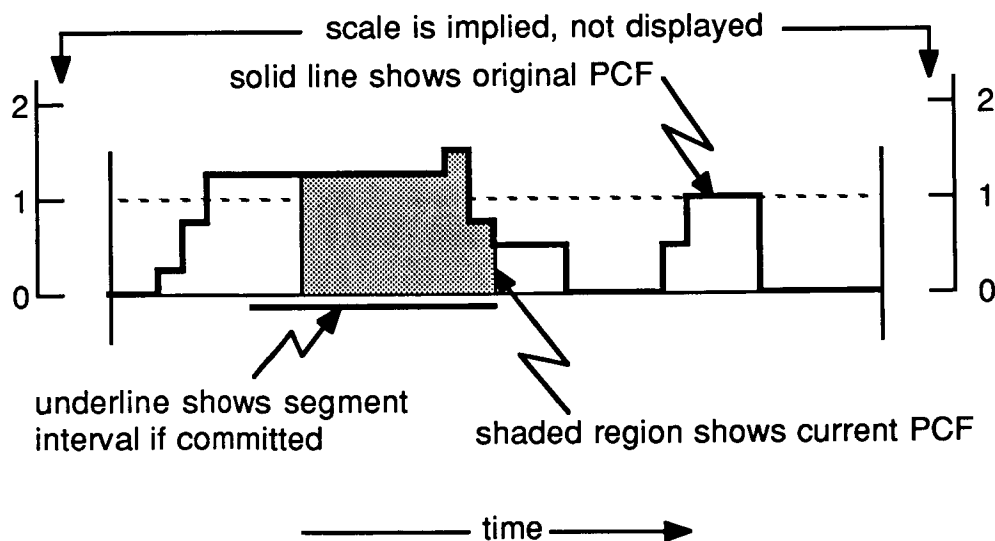


Figure 1 - Suitability function.

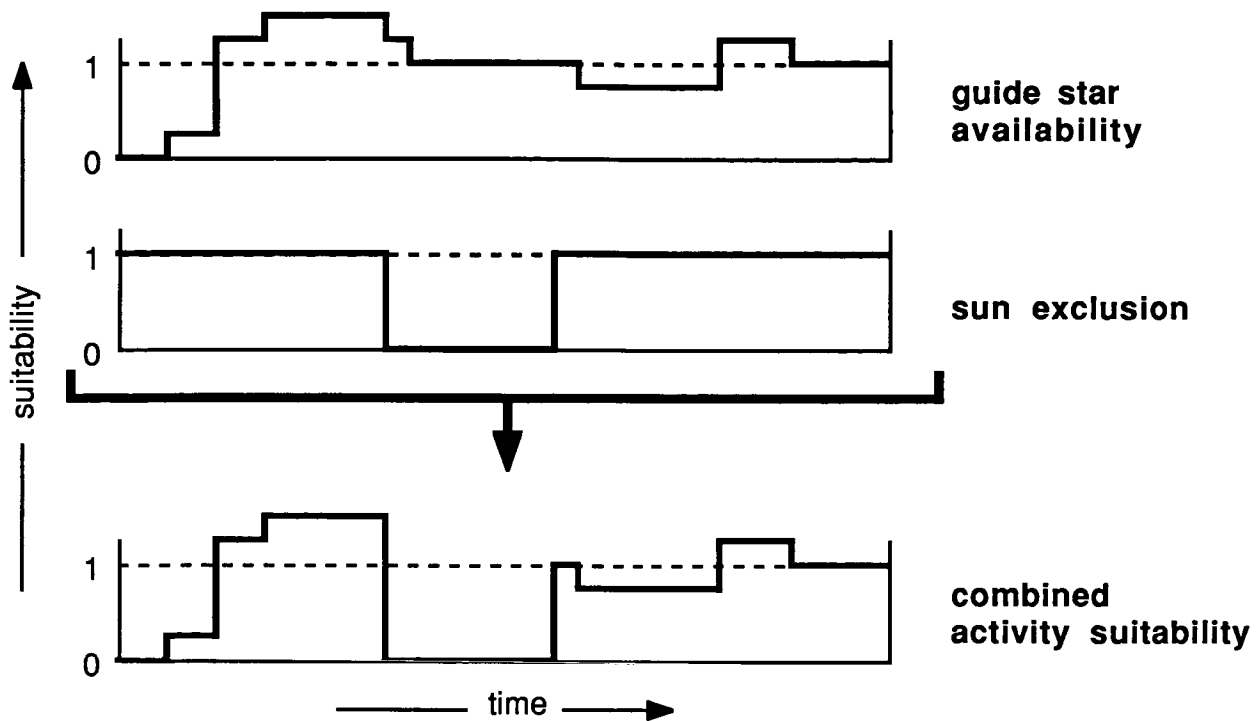


Figure 2 - Combination of two constraints on an activity to form the overall suitability.

The use of suitability functions allows a natural and expressive means to combine the effects of constraints. The overall suitability of an activity is the product of the suitability functions of all constraints (both relative and absolute) attached to that activity. In Figure 2 it can be seen that the strict constraint of Sun avoidance excludes certain times, while the guide star constraint “modulates” the suitability outside the Sun exclusion.

The suitability of a scheduling cluster is the geometric mean of the suitability of all component activities. This requires that all activities have non-zero suitability at that time (since the activities within a scheduling cluster are not necessarily ordered, it must be possible for all to begin at that time).

Constraints are used to restrict the times when it is possible to schedule an activity. The manner in which constraints are propagated in Spike is illustrated in the following example: Let A and B be activities with absolute suitabilities S_A^{abs} and S_B^{abs} derived from their absolute constraints. Let C_1 and C_2 be relative constraints where C_1 expresses a constraint of activity A on activity B and C_2 expresses a constraint of B on A. Given initial values for the suitabilities of activities A and B to be $S_A = S_A^{abs}$ and $S_B = S_B^{abs}$, the constraint processing algorithm evaluates the effect of A on B via C_1 (which depends on S_A) and stores the result in a temporary suitability S_B^{temp} . The suitability of activity B is then updated via $S_B := S_B^{abs} * S_B^{temp}$. Recalling the way suitabilities are defined, any times of zero suitability in S_B^{temp} will result in zero suitability in S_B , other times will result in a larger or smaller suitability depending on the preference expressed in the constraint. The effect of activity B on activity A via constraint C_2 is next calculated in a similar fashion. This iteration continues until there are no more changes in suitability. It can be seen from this that mutual effects of constraints will be propagated onto both activities and that this method is quite

general: no specific assumptions as to the nature or the number of the constraints is required. Constraint propagation is described in more detail in Johnston (1988b).

A **dependency cluster** is a set of activities which are related by relative constraints. They reduce the computational complexity of constraint propagation since the constraint propagation algorithm need only consider clusters in the same dependency cluster.

4.2 Segmentations and Commitments

In Spike, a long range plan is called a **segmentation**. Within a segmentation, time is divided into intervals called **segments**. Segments are simply a convenient means to discretize time, creating time "bins" or "buckets". For long term planning, this allows a significant reduction in the time dimension of the problem without being artificially limiting. The duration of a segment is subject only to the restriction that the time spanned by the segment is greater than an orbital timescale and the duration of activities, i.e. the duration of a segment > 1 day. Segments need not have equal duration, nor need they be contiguous.

A scheduling cluster may be **committed** to a segment, that is, restricted to start during the time interval of the segment, so long as its suitability is non-zero somewhere in the segment. When a cluster is committed to a segment, the cluster suitability function is set to zero outside the segment and the effect of this is propagated automatically. Thus a commitment may further restrict other clusters related via constraints such as precedence or time separation. Note that a cluster may have times of zero suitability within a segment due to the operation of some strict constraint such as Sun avoidance. Any limitations more stringent than that imposed by the segment boundaries are preserved and propagated, thus preserving the maximum scheduling information.

Commitment only requires that the cluster *begin* within the segment; it does not require that the cluster *end* within the segment (which would be unnecessarily restrictive). Multiple clusters may be committed to segments (subject to constraints) and no ordering of clusters within a segment is imposed by the commitment.

A segment may have one or more **segment constraints**. These express limitations segment resources consumed by clusters committed to that segment (e.g. total time, data volume, communications contacts and real time interaction). Given a segment with many clusters already committed, the segment constraint may allow a cluster of short duration to be committed, but prevent the commitment of a cluster of long duration exposures.

4.3 User Interface

Given a problem as complex as HST scheduling, an important aspect of the Spike system is a user interface which facilitates human-machine interaction and rapid comprehension of scheduling constraints, dependencies and commitments.

The main system interface for the Spike software is the Command Hierarchy Interpreter (CHI). This is a general-purpose utility for accessing Spike commands. Commands are organized hierarchically and are accessed by using the mouse and menus. The user of CHI need not ever remember function names or spellings. Context-sensitive help is provided as well as a simple command dependency checker (e.g. command A must be selected before command B).

Within the Exposure Evaluation Tools, the focus of the user interface is the window-oriented planning environment (Figure 3). Time is displayed horizontally, while the suitability function for clusters, activities and constraints can be displayed vertically (refer to Figure 1). This display gives a powerful means to understanding the schedule. The foreground window in Figure 3 displays the suitability functions for 5 related exposures (an acquisition, two science exposures and two

calibration exposures). The background screen displays components of the suitability function for one of the exposures. Functions to access clusters, activities, constraints and the timeline are activated via the mouse and popup menus.

Most of the interaction involves using the mouse and various popup menus: The mouse can be used to zoom in on times of interest. Segmentations, segments, clusters, activities and constraints are all mousable, providing various options which invoke commands. For instance, selecting a constraint's description tells the user the type of constraint, various parameters and the affected activities (e.g. a proposer-specified time separation constraint of 30 ± 10 days between activities A1 and A2).

The user can create multiple planning windows; each showing a different view of the segmentation so far, for example, different time intervals or different sets of scheduling clusters. A change to the plan effected from one window will be reflected in the global database and thus may show up in other windows. The user can also spawn a new, independent segmentation. This is useful for exploring the effects of different commitments in parallel.

Another subsystem that has been incorporated into the Spike system is the Lisp Object State Saver (LOSS, Sponsler 1988). This tool saves memory-resident data structures to an ASCII file. The file can later be reloaded into Lisp memory and the recreated data structures restored to a logically equivalent pre-save state. In Spike, it is used to save partial schedules; a planner using Spike can take a LOSS "snapshot" of memory for use at some later time.

4.4 Operations Concept

In this section we give a brief outline of how the Exposure Evaluation tools can be used in HST science operations. Long term planning begins with the pool of approved proposals in the Pep system. These proposals have also been validated to remove syntactic and semantic errors. Processing of proposals through Pep Transformation, which defines SPSS scheduling data structures, can also be performed at this point, but is not essential for long term planning.

A planner can use the entire accepted proposal pool or any subset (e.g. just high priority and time critical proposals). The relevant proposal information is extracted from the Pep database and transferred to a Spike workstation (see section 7). Next, the planner creates a segmentation timeline, with the time interval of the plan divided into a number of time segments or "buckets". The planner then issues a command to load a set of proposals into Spike. The planner can invoke a command which checks proposals for inherently unschedulable conditions. Such conditions include circular precedence constraints (execute exposure A before B, B before C and C before A) and inconsistent constraints (a proposer specified time window that occurs while the target is too near the Sun for observation). Proposals with inherently unschedulable exposures can be modified, according to policy, to correctly implement the proposer's science objectives.

Exposures are then grouped to form scheduling clusters. One clustering option is to group exposures according to the results of Pep Transformation (either preserving the strict ordering imposed by the SPSS data structures or allowing a more flexible scheme where activities inside SPSS scheduling units are not ordered). In either case, estimates of the amount of resources consumed by a cluster (e.g. time, communications contacts, real time usage, etc.) are calculated. Clustering exposures on the basis of other criteria such as target proximity, instrument usage or relationship to other exposures in the proposal will be explored in future work.

Timespan: 1 year (1990)
36 10-day segments
DRM orbit assumed

Components of suitability function for science exposure S1:

- max 24h separation from C1
- after A by >90d
- before S2 by 60±15d
- roll
- 60m uninterrupted

Exposure suitability functions:

ORIGINAL PAGE IS
OF POOR QUALITY

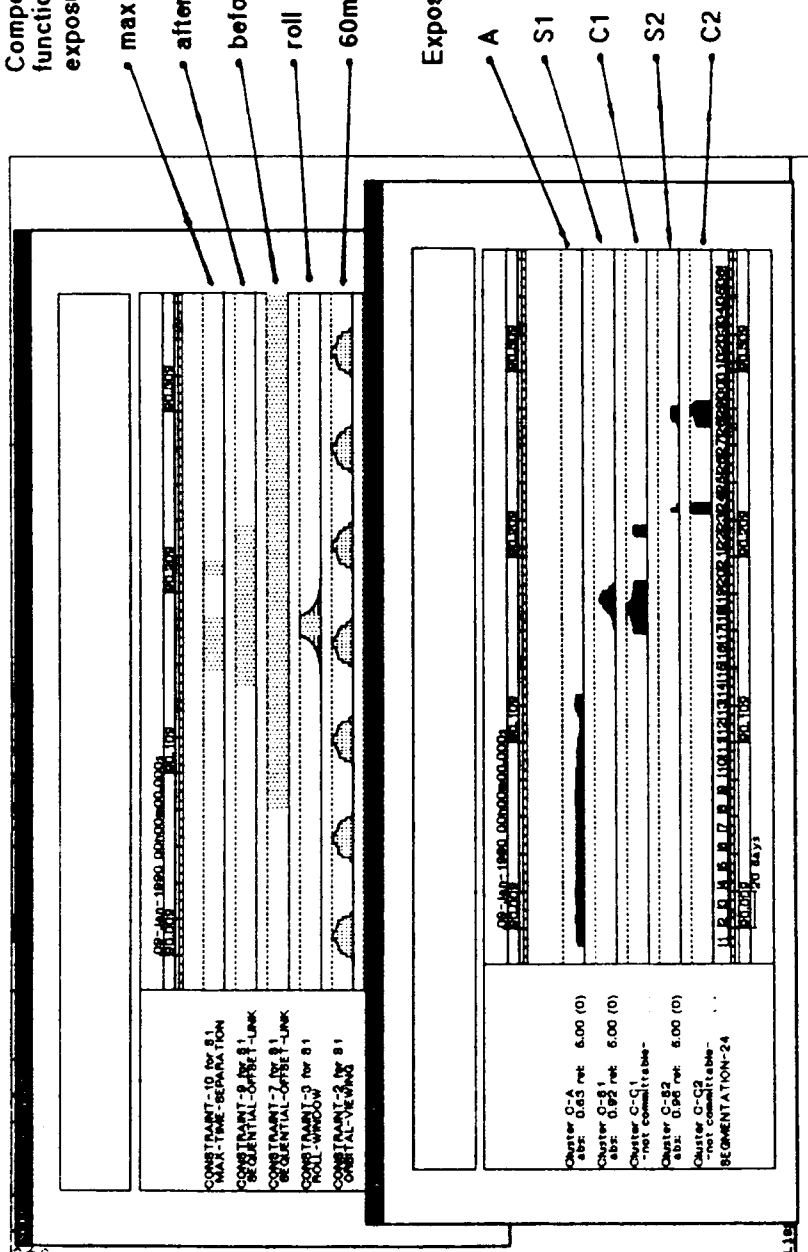


Figure 3 - Exposure Evaluation Tools Timeline Display.

At this point, long term planning can begin, with the goal being to commit each cluster to a segment in what will typically be a year-long or multi-year plan. Several options are available to the planner. Clusters can be committed manually via the user interface, or automatic scheduling software can be invoked (see section 5). The timeline display provides visual information on the commitments and clusters which become unschedulable as the result of other commitments. The schedule is iterated until a suitable solution is found. During the commitment process, the planner can create new segmentations in order to explore different scheduling choices. The state of the system can be saved to a file for use in later planning sessions.

Hierarchical planning fits naturally in this scheme. The planner can create a new segmentation which covers a smaller time interval with finer time resolution in the segments. The commitments from the higher-level plan can be transferred to the more detailed plan for further commitment. Tools to automate hierarchical planning will be developed in future work. When clusters are committed to segments of approximately a week's duration, the commitments can be transferred to SPSS for detailed scheduling.

5. Strategies

Given the large number of observations to be executed by HST in a year, automated tools to schedule the bulk of the observations are clearly necessary. This section discusses three types of automated commitment strategies which have been prototyped: procedural, rulebased and artificial neural networks.

5.1 Procedural Strategies

Two simple procedural strategies are available in the Spike software: most suitable cluster-segment, and most absolute constrained cluster. The first strategy finds the cluster-segment pair with the highest suitability of all clusters in all segments and commits that cluster to that segment. The effect of this commitment is propagated through the relative and segment constraints to find the effects on other clusters. This will usually limit the scheduling choices for these remaining clusters, i.e. the suitability at some times will be smaller (perhaps zero) due to the commitment. Suitabilities are recalculated and the highest cluster-segment suitability of the non-committed clusters is identified. The process is repeated until all clusters are committed or all remaining clusters are unschedulable. This strategy is one implementation of the so-called "greedy" algorithm.

The second algorithm operates in a similar manner, with the cluster which has the smallest time span of non-zero suitability (i.e. most constrained) being committed to the segment with highest suitability at each step.

The advantage of these strategies is the speed of execution. The disadvantage, as is well known in scheduling problems, is that such simple strategies can lead to grossly sub-optimal schedules and an unacceptable number of unschedulable clusters. More flexible strategies which take into account both resource and cluster bottlenecks are required to solve a problem as complex as HST scheduling (Smith, Fox and Ow 1986).

5.2 Rule Based Strategies

In order to provide a more flexible and intelligent approach to scheduling, a rulebased scheduler has been implemented. The rulebase is a **control** layer directing the Exposure Evaluation tools which serves as the **representation** layer. Essentially, the rulebase replaces a human planner making commitments via the window interface. The rulebase chooses what commitments to make and analyzes the results of commitments, while the Exposure Evaluation tools spawn alternative schedules, execute the commitments and propagate constraints. The rulebase system was implemented in KEE (a product of IntelliCorp).

Communications between the two layers is through a schema in the rulebase. For each segmentation (partial schedule) in the representation layer, there exists a corresponding schema in the control layer. The schema contains a number of slots which hold summary information about the segmentation, e.g. segmentation name, most suitable cluster and segment, most absolute constrained cluster, highest priority cluster, unschedulable clusters, etc. All reasoning about commitments uses this summary information. When a commitment is made the representation layer creates a new segmentation and the control layer creates a new schema. A method is executed which populates the schema with summary information. Both the schemas and the segmentations can be thought of as a tree where each node is a partial schedule.

The search strategy conducts a "best-first" search based on the A* algorithm. Two classes of rules were used: search and commitment. The search rule class identifies which one partial schedule is most promising (in some sense, see below). The commitment class of rules decides for the most promising node, what is the best commitment to make (e.g. highest priority cluster, most absolute constrained cluster).

Definition of the "best" or "most promising" partial schedule is of central importance to this strategy. Work to date has primarily used the sum over all clusters of the highest suitability in any segment. This allows the rulebase to identify poor solutions (since the suitability of some clusters becomes low), but does not provide enough discrimination between schedules before poor solutions are generated. This is largely due to the fact that the summed suitability neglects the effects of segment constraints until a cluster becomes totally unschedulable. In other words, it is focusing too closely on scheduling clusters in suitable segments without considering the effects on the diminishing resources of the segments themselves. Smith and Ow (1985) have addressed the similar problem of task versus resource based views in factory scheduling. We have also made preliminary investigations of using the rulebase to identify resource bottlenecks in order to schedule critical clusters first.

A typical commitment rule might be:

```
For the best partial schedule
if   there is a high priority exposure
    and it is the most absolutely constrained exposure
    and this rule hasn't been applied to this partial schedule
then commit the exposure to its most suitable segment
    populate the schema with segmentation information
    add this partial schedule to the list of open schedules
```

This example is paraphrased from the Kee code, however implementing a natural language tool to take the above and translate to executable code is not a difficult task. In the event that more than one commitment rule is instantiated for the current best node, the rule with the highest weight is chosen. More sophisticated strategies for choosing among competing commitment rules are under investigation.

5.3 Artificial Neural Network Strategies

An artificial neural net has been implemented as another approach to automated scheduling. Neural nets are based on models of how biological "computers" work and have been applied to a variety of problems including pattern recognition, classification, memory, and speech understanding (see Tank and Hopfield 1987 for an introduction to neural nets).

In SPIKE scheduling, a version of the Hopfield neural net model is used. The network can be considered as a rectangular matrix, where the columns represent segments (time) and the rows represent scheduling clusters. A particular network element (neuron) represents the potential

commitment of a particular cluster to a particular segment. When the network has relaxed (found a solution), a neuron in the "on" state signifies a commitment of that cluster to that segment. For each neuron, a bias term expresses the absolute suitability of that cluster-segment combination, with the bias term proportional to the suitability in that cluster (or a large inhibitory bias if the suitability in that segment is zero). Several factors determine the connections strengths between neurons: All neurons in the same column (same segment) are connected to express the segment constraints. All neurons in the same row (same cluster) are connected so that a cluster committed to a segment will inhibit the commitment of that same cluster to another segment. The effect of the commitment of cluster A1 to segment S1 on the commitment of cluster A2 to segment S2 is implemented as a connection between the corresponding neurons. The strength of the connection (weight) is determined using suitabilities derived from the relative constraints relating A1 and A2. Additionally, there is a set of "guard neurons", one for each row (cluster) which apply a bias to force each cluster to be committed to some segment. Without the guard neurons, the network could converge to solutions where many clusters were not committed to any segment. However, the addition of guard neurons also breaks the symmetry of the basic Hopfield model so that the network is no longer guaranteed to always converge to a solution. With the network connections established, the neural network tends to frequently find "good" solutions (most suitable cluster/segment commitments) very quickly. When the network fails to converge it can be stopped after a fixed number of neuron state changes and then restarted. In this way it is possible to quickly generate and evaluate a number of partial or full schedules. The neural network approach appears particularly promising for replanning: the network does not need to be rebuilt since changing only the neuron biases is sufficient to indicate which activities have been executed. It also offers the potential for implementation on parallel hardware, which would provide even more dramatic performance improvements.

6. Development Methodology

The software described in this paper has been developed using "artificial intelligence" tools, including Lisp, object oriented programming, expert systems and artificial neural networks. The use of a Lisp software development environment on a workstation (Texas Instruments Explorer and Apple Macintosh) provided an unparalleled development and implementation environment.

A rapid prototyping methodology is a key component of our approach, since many requirements of the problem are not understood in detail - the HST scheduling problem is orders of magnitude more difficult than that found at any other observatory. Examples of areas requiring substantial investigation include the relative importance between various constraints, the choice of scheduling strategies in various situations, metrics for schedule quality, replanning from schedule disruptions and display tools. Our development approach is to quickly provide the users an initial set of tools with fundamental scheduling capability. This testbed will allow users and developers to explore the problems together, better understand the requirements and implement improvements quickly and efficiently (see Agresti 1986 for more on rapid prototyping).

Most of the system is implemented in Common Lisp. Currently we are using the Flavors object system but we expect to migrate to the Common Lisp Object System when it becomes available. The graphics and user interface utilities are based on IntelliCorp's Common Windows. The use of expert system shells such as KEE (from IntelliCorp) or ART (from Inference) to control scheduling decisions is under investigation. Careful attention has been paid to standardization and portability: portions of the system have been used on TI Explorer, Symbolics, Vax, Macintosh and Sun computers. Although development takes place on Explorers and MacIntoshes, we are investigating the utility of other machines for operational use including general purpose workstations (such as Sun computers) or hybrid machines (such as the Macintosh II-micro Explorer).

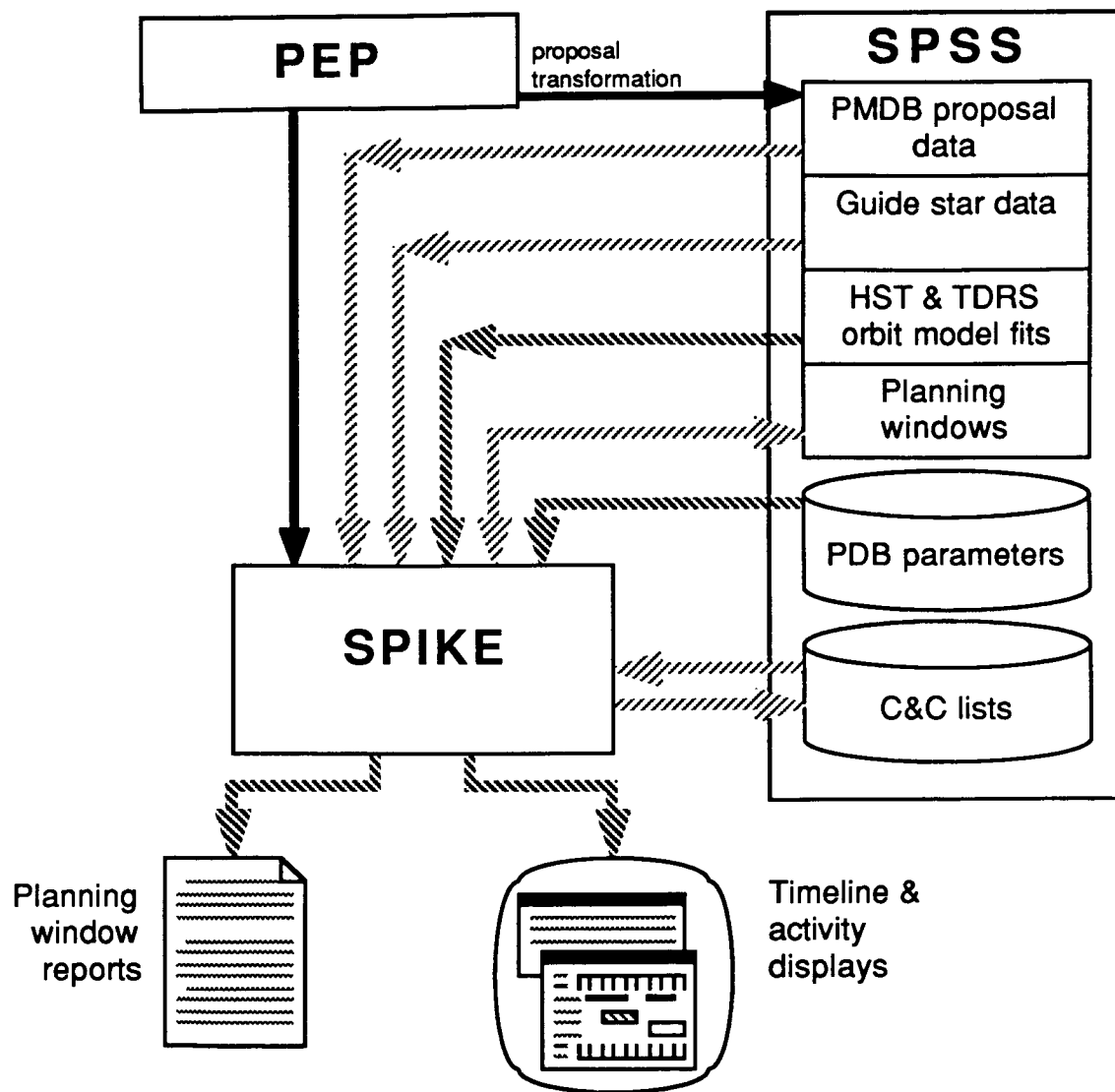


Figure 4 - Spike Interfaces. Solid arrows show existing interfaces, dark hatched arrows show interfaces to be implemented for Exposure Evaluation Tools, while light hatched arrows show interfaces planned for later phases. Interfaces to SPSS (e.g. Guide Star Selection System) are not shown.

7. Interfaces

The Spike system has interfaces to two other systems: Pep and SPSS (see Figure 4). Currently, the Pep interface has been partially implemented. Interfaces to the SPSS system will be added as development continues.

As noted earlier, the Pep system contains proposal information including the proposer's target list, exposures and their interrelationships, and the initial SPSS data structures generated by the Pep Transformation software. Migration of this information from the Pep relational database to the Spike system is a two step process: First, a general purpose database report writer is used to extract the proposal information into a file, formatted as calls to Lisp functions. In the second step, a Lisp program reads the database report file and creates a second file, again writing calls to Lisp functions. When loaded into the Spike system this file creates data structures such as targets, constraints, activities and clusters. The first step is simply an extraction of the information from a relational database, while the second step performs the mapping between the Pep and Spike data structures. For example, binary links between exposures are stored as tuples in a relation in the Pep database. These links are mapped to different Spike constraints depending on the type of the exposure link (acquisition, precedence, conditional, etc.).

Files are transferred from Pep to Spike using Decnet or TCP/IP and can be initiated from either system. Currently, extraction of Pep information must be initiated on a computer running the Pep system. The second processing step can be performed on either a Pep or Spike computer. We plan to implement remote procedure calls so that all processing can be initiated from the Spike workstation.

Initially, a unidirectional interface from SPSS to Spike will be implemented in order to provide Spike with HST and TDRS orbit models and HST Project Database parameters relevant to long range scheduling. The interface will be extended to provide bi-directional transmission of scheduling windows and planning data structures, e.g. transmission of medium and short range plans from Spike to SPSS for detailed scheduling and the feedback of these results into the Spike plans. The interface will be patterned on the Pep-Spike interface as the relevant SPSS information resides either in a relational database or in disk files.

8. Discussion

The Spike Exposure Evaluation Tools provide a powerful means for long range scheduling of HST observations. Essential aspects in reducing this to a tractable problem include the representation and propagation of constraints, clustering exposures to reduce the number of entities scheduled, discretization of time to reduce the number of places examined and a user interface which displays the relevant information in a succinct manner.

The problems faced in scheduling HST observations are common to other telescopes and the Spike system has been designed with this in mind: The concept of suitability functions as a means of expressing constraints and constraint satisfaction is a general one. Likewise, propagation of the effects of constraints is not tied to the nature of the constraint or scheduling problem. Indeed, this approach should be applicable to many classes of scheduling problems outside the realm of telescope scheduling.

Traditionally, little emphasis has been given by observatories to integrated, efficient scheduling programs: For the most part, ground-based telescopes are scheduled manually (perhaps with limited software assistance), often by granting blocks of time (days or weeks) to observers who execute the observations. More sophisticated scheduling techniques can be found in some space-based telescopes. These schemes can limit the scientific efficiency of an observatory: repeated

observations of short duration over a long timespan are very difficult to accommodate in the block time scheduling. An observation requiring outstanding atmospheric or orbital conditions may find useless time that would be more than adequate for other, less demanding observations. Simultaneous observations of a target from several observatories only exacerbates the problem. Given the high oversubscription rate of all major astronomical facilities, an increase in efficiency can be very important. Concurrent with the increased abilities of scheduling software, there is a growing awareness of the need for automated planning to increase scientific productivity (Johnston 1988a). Plans for the University of Texas-Penn State 8m Spectroscopic Survey Telescope (SST), the European Very Large Telescope (VLT) and Space Station Science and Applications Information System (SAIS) call for increased scientific return through sophisticated scheduling and reactive replanning. We have made preliminary investigations in scheduling other observatories with the Spike software (Johnston 1988c).

We would like to thank Hans-Martin Adorf (ST European Coordinating Facility), Robert Jackson (STScI), Don Rosenthal (NASA Ames), Tom Sherrill (Lockheed), Dave Skillman (NASA Goddard), Steve Smith (Carnegie Mellon University) and Monte Zweben (NASA Ames) for stimulating conversations on planning and scheduling problems. We also express appreciation to Texas Instruments for the loan of a TI Explorer.

References

- Agresti, W. 1986, *New Paradigms for Software Development*, IEEE Computer Society Press.
- Hall, D., ed. 1982, *The Space Telescope Observatory*, NASA CP-2244.
- Jackson, R., Johnston, M., Miller, G., Lindenmayer, K., Monger, P., Vick, S., Lerner, R. and Richon, J. 1988, "The Proposal Entry Processor: Telescience Applications for Hubble Space Telescope Science Operations", *Proceedings of the 1988 Goddard Conference on Space Applications of Artificial Intelligence*.
- Johnston, M. 1988a, "Automated Telescope Scheduling" in *Coordination of Observational Projects*, Cambridge University Press.
- Johnston, M. 1988b, "Reasoning with Scheduling Constraints and Preferences", in preparation.
- Johnston, M. 1988c, "Automated Observation Scheduling for the VLT", in preparation.
- Miller, G., Rosenthal, D., Cohen, W. and Johnston, M. 1987, "Expert Systems Tools for Hubble Space Telescope Observation Scheduling" in *Proceedings of the 1987 Goddard Conference on Space Applications of Artificial Intelligence and Robotics*, reprinted in *Telematics and Informatics*, 4, 301-311.
- Rosenthal, D., Monger, P., Miller, G. and Johnston, M. 1986, "An Expert System for Hubble Space Telescope Ground Support" in *Proceedings of the 1986 Goddard Conference on Space Applications of Artificial Intelligence and Robotics*.
- Sherrill, T. 1987, Lockheed Engineering Memorandum, private communication.
- Smith, S., Fox, M. and Ow, P. 1986, "Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems", *AI Magazine*, Fall 1986, p45.
- Smith, S. and Ow, P. 1985, "The Use of Multiple Problem Decompositions in Time Constrained Planning Tasks", *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*.
- Sponsler, J. 1988, "Lisp Object State Saver (LOSS): A Facility Used to Save Partial Schedules for the Hubble Space Telescope" in *Proceedings of the 1988 Goddard Conference on Space Applications of Artificial Intelligence*.
- Tank, D. and Hopfield, J. 1987 "Collective Computation in Neuronlike Circuits", *Scientific American*, December 1987, pp 104-114.

The Proposal Entry Processor: Telescience Applications for Hubble Space Telescope Science Operations

Robert Jackson¹
Astronomy Programs, Computer Sciences Corporation

Mark Johnston
Space Telescope Science Institute²

Glenn Miller¹, Kelly Lindenmayer¹
Astronomy Programs, Computer Sciences Corporation

Patricia Monger, Shon Vick, Robin Lerner, Joel Richon
Space Telescope Science Institute²
3700 San Martin Dr.
Baltimore, MD 21218

The Proposal Entry Processor (PEP) System supports the submission, entry, technical evaluation review, selection and implementation of Hubble Space Telescope observing proposals. This paper describes the PEP system, concentrating on features which illustrate principles of telescience as applied to the HST. These principles are applicable to other observatories, both space and ground based.

The PEP proposal forms allow a scientist to specify scientific objectives without becoming needlessly involved in implementation details. The Remote Proposal Submission System (RPSS) allows proposers to submit proposals electronically via Telenet, SPAN and other networks. RPSS performs syntax and semantic checks on proposals. PEP uses a fourth generation database system to store proposal information and to allow general queries and reports. The Transformation subsystem uses an expert system written in OPS5 to cast a scientific description of an observing program into parameters used by the planning and scheduling system. The TACOS system is a natural language database which supports the proposal selection process. Technical evaluations for resource usage and duplicate science are performed using rulebased systems.

¹ Staff member of the Space Telescope Science Institute

² Operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration

1. Introduction

The PEP System at the STScI is an interface between the HST user and the planning and scheduling software. The purpose of this system is to accept a high level description of an astronomical observing program and to produce from this the parameters necessary for the planning and scheduling software. Additionally, PEP provides tools for technical evaluation and selection of observing proposals. Traditionally, the processes of solicitation, selection, evaluation and implementation have been largely manual. The tools provided by the Pep system provide a novel approach and may be useful as a model for proposed ground and space-based observatories.

This paper describes the Pep system and how it assists in proposal processing. Section 2 describes from a general point of view the process of soliciting and implementing an astronomical observing program and describes some of the high level requirements of this process. Section 3 describes the STScI proposal forms which allow an astronomer to describe an observing program at a high level, without being needlessly burdened by implementation details. Section 4 examines the architecture of the Pep system and provides an overview of the major subsystems: Entry, Evaluation, Transformation and Time Allocation Committee support. Section 5 discusses the PEP Remote Proposal Submission System in detail. The last section discusses PEP in the context of Telescience and applications to other observatories.

2. The Observing Process

Consider first the observing process, from the formulation of a scientific problem through the collection of data: An astronomer poses a question, decides what observations would answer the question, identifies a telescope and instrumentation which are capable of obtaining these observations, plans the observations, and finally executes the observations and collects the data. For example, an astronomer may ask "what is the distribution of stellar ages?" Knowing that lithium abundance is correlated with age would lead to observing red giant stars to obtain lithium spectral line profiles. An observing proposal for time on a telescope with a suitable aperture and spectrograph would be submitted. Prior to observing, a sample of red giants stars would be picked, and parameters such as wavelength ranges and exposure times would be determined. Lastly, observations are made, perhaps adjusting the program to compensate for changes in weather, instrument performance, etc.

This is, of course, a very brief and idealized sketch of what is often a long and complicated process! The essential point is that during this process, the expression of the observing program must undergo a number of transformations, from general descriptions to specific implementation details.

For many observatories, there has been no need to explicitly enumerate these observing steps as one person, the observer, was responsible for most aspects of the program. The transformation from general program to specific instrument and telescope operations was implicitly performed by the observer, often "on-the-fly". This is particularly true for classical, ground-based telescopes, and to a lesser extent for space-based telescopes.

There are two reasons why it may be necessary or desirable to consider the observing process more explicitly: First, the operation of some telescopes is so complex that an observer has neither the time nor the motivation to acquire the necessary expertise to implement the program singlehandedly. HST serves as a good example of this. The input to the planning and scheduling software requires a detailed understanding of the HST, orbital conditions, and the software data structures. The ground system requires that most observations are planned in detail weeks in advance of execution. Observers are not in continuous communications contact with the HST. Telescope and instrument operations must be carefully examined to ensure the health and safety of the spacecraft.

A second reason is that important gains in scientific efficiency can be achieved. Johnston (1988a,b) has addressed the need for automated scheduling of ground and space based telescopes. Interleaving of observations from different programs, instead of block scheduling of time can lead to increased scientific return by minimizing instrument changes and calibrations, and accommodating programs which require short observations over long time periods. Many observatories offer "service observing" where observatory staff members execute observations for the proposer. At ground based observatories, it is all too common for a program requiring excellent atmospheric conditions to be executed during a time of mediocre seeing, while a program with less stringent requirements happens to occur during the best seeing conditions.

Computer science, in particular, the field of artificial intelligence (AI) has matured sufficiently that we can begin to tackle these problems. Although there is still much debate over whether computers exhibit any form of intelligence, it has been clearly demonstrated that computers are solving problems which formerly required highly trained humans and that traditional (non-AI) computing paradigms are inadequate to deal with these tasks.

The thesis of this section is that the observing proposal form and proposal processing must be considered in view of the above. A proposal is not simply used by an observatory to select observers. The proposal form must contain sufficient information that the observatory can enhance the observing process and increase the scientific productivity of both the observatory and the observer. This includes proposal selection, evaluating proposals for feasibility and efficiency, implementation and data archiving.

The process of elaborating the scientific program can be thought of as progressing from *Asking Questions* to *Identifying Data Needed* and finally to *Specifying Instrument Activities*. Before describing the "Astronomer oriented" vocabulary by which the HST user characterizes their programs, it is useful to consider various possible vocabularies. A user might describe their programs at three levels of abstraction: Answers, Data, and Spacecraft Activities

A scientist could simply ask a question, e.g., "What is the relationship between mass of a galaxy and its central velocity dispersion?" Software would determine the data required and either locate the data in archives or else determine the telescope activities needed to obtain the data. However, such software does not yet exist and is beyond the state of the art of AI today.

The next level of abstraction would be for the user to specify the data they wanted. The characteristics of astronomical data include: spatial range, spatial resolution, spectral range, spectral resolution, time range, time resolution, and signal to noise ratio for the flux. The software would search the archives and determine the best choice of spacecraft activities which would generate the desired data. The software faces a much more limited range of possible inputs than when answering a more general question, but it would still need detailed knowledge about the spacecraft capabilities and how to obtain the required data. Such a software tool could be built with today's technology, but it would be a significant undertaking.

Instead of having the proposer specify the questions or the data, the proposer would specify the spacecraft activities needed to obtain the desired data. Here, an analogy with computer languages becomes relevant. A programmer can work in assembly language or in a high-level language like Fortran or Pascal. Similarly, in specifying the spacecraft activities, the proposer could specify the actual spacecraft command loads on a timeline or else could specify very high-level commands in a "friendly" language.

When specifying detailed spacecraft activities, it is all too easy to request physically impossible, unpermitted, or internally inconsistent activities. This problem is much less likely to occur when specifying higher level commands. The complexities of modern spacecraft and of spacecraft

operations require detailed knowledge which is costly to acquire. The syntax used in specifying the activities must be validated to assure that only legal syntax was used and that the activities are feasible.

3. HST Proposal Forms

Users of ground based telescopes and previous spacebased telescopes have traditionally been granted specific time periods in which they control the telescope in real time. The proposal forms for these facilities have only contained information needed for proposal selection and for providing necessary staff and hardware resources at the telescope. The forms have not contained a detailed description of exactly what observations are to be done. The situation with HST is quite different. Real time communication with HST is available about 20% of the time, due to HST being in a low earth orbit and sharing the 1 MHz Tracking Data Relay Satellite data link. Thus the proposer must specify on the proposal forms all the information needed to execute the observations. An analogous situation for a ground based telescope would be if the astronomer had to write a computer program which would command the telescope and perform all the desired observations.

Planning and scheduling of HST observations is currently performed with the Science Planning and Scheduling System (SPSS) of the Science Operations Ground System (SOGS) developed by TRW. SPSS requires its input to be in a syntax and form which is set by the design of the SOGS software and internal data structures. A scientist would need to have a great deal of specialized knowledge about the internal operations of SOGS in order to properly describe his series of desired exposures in the SOGS syntax. The burden on the scientist would be much too great, and the probability of error in the specification of the exposures would be much too high.

To meet this limitation of SOGS, the STScI developed the HST proposal forms, described in the *Call For Proposals* and *Proposal Instructions* (1985), with the following goals:

- Be oriented towards the user community - easy to understand, and concise and logical in the amount and sequence of data requested

- To accommodate both simple and sophisticated observations

- To allow the proposer to specify what data should be collected without becoming needlessly encumbered by telescope and instrument specific details

- To allow data entry by entry clerks directly from the submitted forms with a minimum amount of training.

The following sections describe the Proposal Forms, i.e., the Coverpage and General Form, the Targets Lists, and the Exposure Logsheet.

3.1 Coverpage and General Form

The Cover Page is an "executive summary" of the proposal, and contains the proposal title, scientific category, the principal investigator, the number of targets to be observed, the amount of exposure time requested, a scientific abstract, and the amount of funding requested.

The General Form largely expands on the Coverpage information. There are General Form sections listing all the investigators and their address. There are other sections in which the proposer describes in detail the scientific justification of the project, why HST is needed, why special scheduling or calibration requirements are necessary, what the data analysis plans are, etc.

The information on these forms are used primarily in the proposal selection process and for contacting the proposers, but they also provide a written description of the proposer's intended observations. The only limitation on the content of the forms is that PEP cannot enter graphical data or non-alphanumeric characters, e.g., Greek letters.

3.2 Target Lists

The HST proposal forms allow for three categories of targets: Fixed, Solar System and Generic, each with its own Target List. This division is necessary since the specification of target position is different for each class.

Fixed targets are defined by a specific position on the sky. The proposer may specify a particular right ascension and declination (along with uncertainties), or an offset in coordinates from another fixed target. Specification of extended or area targets (e.g. nebulae, galaxies) is also possible.

The positions of Solar System targets are specified in one of a number of ways, including standard names (e.g., Jupiter), orbital elements (e.g., a new comet), or positions relative to other solar system objects (e.g., satellite or planetary surface features). The positions can also be restricted to specific time intervals or periods when certain planetary events occur (e.g., maximum elongation of a satellite). The Moving Object Support System (MOSS) was developed by JPL and converts these position and time specifications into a time series of position vectors which are used by SOGS in scheduling the observations.

Generic targets provide added flexibility to the proposer. These targets are identified by general target characteristics or broad locations in the sky. Examples of generic targets include targets of opportunity (nova, supernova, comet, etc.) or certain types of targets in large regions of the sky (e.g. any field within 10 degrees of the north galactic pole.) Generic targets are useful when it is unduly restrictive or impossible to select a specific target at the time of proposal submission.

Each Target List requests a target name, target description, anticipated HST acquisition problems and target brightness data. Target names and description are important in the construction of useful astronomical archives and in understanding the proposer's intent. The STScI Proposal Instructions give detailed guidelines for naming and describing targets. Target brightness data is requested so that exposure times can be independently verified and adjusted to compensate for orbital conditions (e.g. scattered earthlight)

3.3 Exposure Logsheet

The Exposure Logsheet provides a powerful mechanism for expressing the observations to be done at the positions specified on the Target Lists. It provides both a simple to use form for the common types of observations, and yet a powerful means of expressing complicated programs with many interdependencies. This form ties the exposure information to the target list information and contains all the information provided by the user describing the spacecraft activities desired.

For simple observations, the user specifies on this form the *Target Name, Configuration, Operating Mode, Spectral Element, Entrance Aperture, Flux Reference Number, Number of Exposures, and Exposure Time*. For more complicated observations, the *Optional Parameters* allow the user to change the instrument settings from the default values. This is especially useful for onboard target acquisition modes, where the nature of the field determines the best set of search parameters. The use of default settings allows the HST user to ignore those settings which are not relevant to his needs and allows STScI to use the best values based on recent experience without having to alter the contents of the Exposure Logsheet whenever an improved value is determined.

The *Special Requirements* section of each Exposure Logsheet line allows the user to specify:

Relationships between exposures, e.g.,
Early Acquisition For <line>
Same Orientation For <line> As <line>
Calibration For <line>
Real Time Analysis For <line>
After <line>
Sequential <lines> Within <time>

Additional properties of an exposure, e.g.,
Position Target <X,Y>
Spatial Scan
Critical Observation
At <date> +/- <range>
Dark Time

Branches and conditional exposures, e.g.,
Branch To <line> If <condition>
Conditional If <condition>
Select <number> of <lines> Or <lines> ...

Each exposure on the Exposure Logsheet is labeled with a line number. The <line> in these Special Requirements refers to a single line number or a range of line numbers.

These *Special Requirements* must be described using syntax which is listed in the Proposal Instructions. The syntax limitations allows the user's needs to be processed by software automatically and without any human interpretation.

The Exposure Logsheet provides several constructs which allow the user to express a set of exposures in a succinct fashion. The *Sequence Definition or Usage* column can be used to Define subroutines of exposures and then to Use such subroutines. In the Define lines, some of the columns can have placeholder symbols, which get substituted with specific values from the Use lines. In the *Special Requirements* column, the user can state

Do For Targets <numbers>

or

Repeat <lines> Every <time> For <number> More Times

to either execute an Exposure Logsheet line for a number of targets or to execute a number of lines at the stated interval.

Not only does this eliminate bulk and repetition, it can also make the proposer's intent more clear to both humans and software. This subroutine or do-loop ability is functionally similar to block structured computer programs.

There are two additional forms which are used in the small number of proposals where additional information is needed. The Proper Motion/Parallax Form is used to specify the apparent motions of fixed targets when these motions are significant (e.g. could affect target acquisition). The Scan Data Form is used when the HST Pointing Control System scanning capability is used, i.e., when the Spatial Scan special requirement is stated.

While the forms and syntax are more complicated than proposal forms for ground based telescopes, they provide a compact and expressive representation of the series of activities and decisions desired by the user. The information required for software to execute an observing program with a ground based telescope would be similar in volume to that required for HST.

4. PEP System Overview and Design

The PEP system was designed to support the Entry, Evaluation, Selection, and Transformation of HST observing proposals as described in the *ST Proposal Entry Processor System Requirements* (1987). The process by which proposal information flows through the PEP system is illustrated in the following diagram.

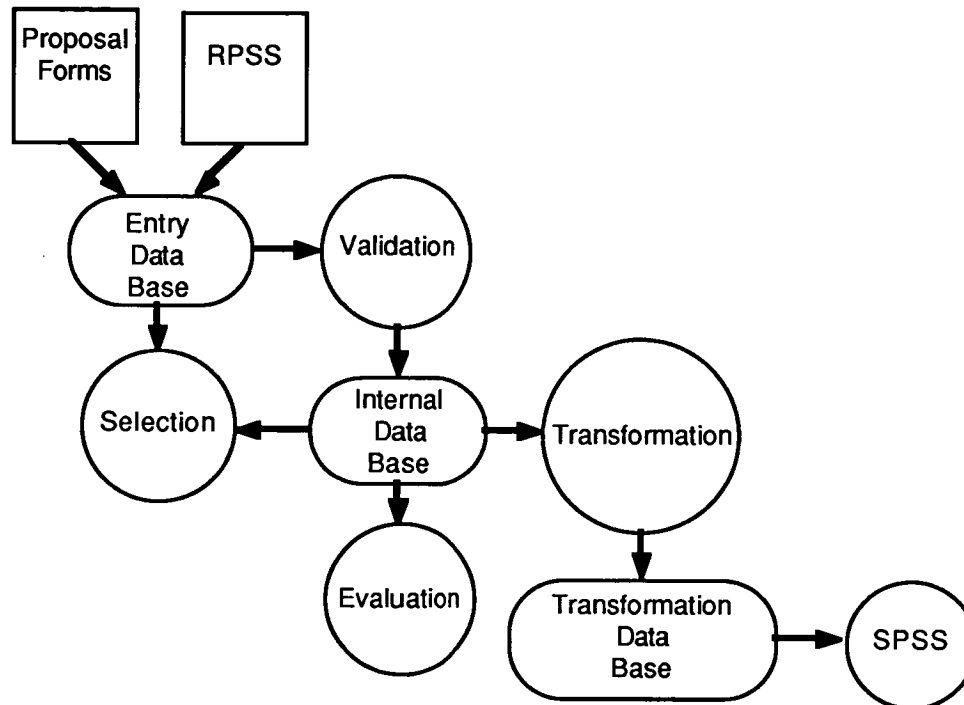


Figure 1 - Overview of PEP System

The Entry subsystem consists of RPSS (described in Section 5) and other software tools and provides entry and editing capabilities for the Entry Data Base (EDB) and the Validation function which populates the Internal Data Base (IDB). The Evaluation subsystem is a set of tools which uses IDB information to check feasibility problems, find duplicate exposures, etc. The Selection subsystem uses EDB and IDB information to aid the proposal selection process. The Transformation subsystem converts the IDB information into the SPSS data representation which is stored in the Transformation Data Base (TDB) for later transmission to SPSS.

4.1 PEP Development Environment

The PEP software was developed in a rapid prototyping environment where the developers had as much understanding of the system requirements as the users had. This combination of a rapid, iterative code enhancement cycle and a high level of domain expertise by the developers provided a working PEP system quickly and economically.

The Entry Data Base (EDB), Internal Data Base (IDB), and Transformation Data Base (TDB) are relational data bases in a Britton-Lee BL700 data base machine. The interface between the BL700

and the VAX's is provided by Signal Technology's Omnibase, a fourth generation language tool. The extensive querying capability provided by Omnibase is very useful in an environment where not all the operational requirements are known in advance.

All the code was developed in a VAX environment under the VMS operating system, i.e., the same hardware environment as SOGS. The computer languages were chosen to match the needs of the subsystem. The Entry subsystem is largely written in C and uses the flexible data structures available there. A large portion of the Transformation subsystem was written in OPS5, a rule based production language ideal for handling large numbers of special cases. The core of the Selection subsystem was written in Common LISP, a language well suited for natural language processing. The ability to choose the best language for the task has help to quickly create the software tools needed for each subsystem.

4.2 PEP Entry Subsystem

The Entry Subsystem takes the user's information from the proposal forms or RPSS files (described in Section 5) and enters it into the Entry Data Base (EDB). The Entry Subsystem was designed to be robust and to allow entry of any alphanumeric information on the proposal forms. It was also designed to be easy to use and require a minimum of training for the entry clerk.

The ease of use was met by having the clerk enter data into terminal screen templates which closely resemble the proposal forms. The robustness was met by having the entry tools allow any alphanumeric information on the form to be entered and stored in the database. The ability to enter the entire contents of the paper form means that illegible characters or incorrect syntax will not halt the entry process.

Since the EDB information is in the form of free text and thus can contain either valid or invalid syntax, there is a Validation tool which verifies that the user has followed the Proposal Instructions and has used only legal syntax. For example, Validation will verify that the user has requested a filter which exists for the desired instrument configuration and operating mode. The Validation tool takes the free text input, generates error messages describing any illegal syntax, and populates the Internal Data Base (IDB) relations. Whereas the EDB relations contain free text and are organized along the lines of the proposal forms, the IDB relations contain specific numeric or character values organized in a hierarchy which describes exposures and targets, their properties, and their interrelations. If the Proposal Forms and syntax is a computer language, then Validation is a compiler.

The EDB provides an insulating layer between the proposal forms and the IDB representation of the proposal information. Should the proposal forms be changed, none of the subsystems which get their input from the IDB will be affected by the change. The only changes would be to the EDB, to the Entry tools, and to Validation which takes the EDB information and populates the IDB.

The screen based entry tools can also be used to search the EDB for information in a specific proposal. There is also a very powerful interactive query language and report writer which can be used to make complex queries and format the results of the queries.

Another requirement of the PEP system was to provide proposal security and change tracking. The security is provided by allowing only people in certain lists the ability to read or edit proposal information. When a user edits a line on a form, a record is sent to a proposal history relation which tracks who changed what and when. There is also a Signoff facility which allows pending changes to be accepted or rejected by staff having this level of access to the system. With these security features and the backups of the EDB, PEP is able to limit access to proposal information and to recover from erroneous user changes.

This subsystem has been extensively used in entering the 307 Guaranteed Time Observer proposals, the approximately 250 Orbital Verification and Science Verification proposals, and more than 100 General Observer proposals. Since its initial delivery, the PEP Entry Subsystem has been enhanced to deal with additional forms and with new syntax, and has proven to be a successful and operational system.

4.3 PEP Evaluation

The purpose of the PEP Evaluation subsystem is to evaluate proposals and assist STScI staff in reviewing the technical feasibility of proposals. This includes identifying possibly redundant exposures, spacecraft resources consumed by the exposures, and impossible to implement exposures. The Evaluation Subsystem has three general functions: Duplication Checking, Resource Usage, and Feasibility.

4.3.1 Duplication Checking

Since HST time is such a scarce resource, it is important to check for the possibility of duplications with either previously executed and archived exposures or concurrently proposed exposures. However there are more than 10,000 exposures scheduled each year and the number of possible pairs of "duplicate science" exposures is very large, of order N^2 . The purpose of the Duplication tool is to identify a small number of possible duplications for more detailed evaluation by a human scientist. No proposed exposure is rejected without a scientist evaluating the significance of, or the need for, what appears to be a scientific duplication. There could be situations where duplicate science is necessary, e.g., confirming suspected time variability of a phenomenon.

Scientific duplication is assessed both by positional similarities and by instrument usage similarities. Due to the wide field of view of certain HST detectors and to the uncertainties in the coordinates provided by the users, there are degrees of position matching criteria. Similarly, since equivalent scientific information can be obtained with different instrumental setups, there are several degrees of instrument matching.

The Duplication tool is written in C, OPS5, and the data base query language, IQL and is described in Jackson (1987). The position matching and instrument matching code is written in OPS5, which allows for rapid development and modification of the algorithms to meet new input syntax or new definitions of "duplication".

4.3.2 Resource Usage

Individual proposals can vary quite widely in the resource overheads needed to execute the same total exposure time. For example a one second HRS exposure in the middle of a string of other HRS exposure on the same target would take one second to execute. But a lone one second WFPC exposure could take 10 minutes to slew the spacecraft, 12 minutes to acquire guide stars, 1 second to expose the CCD, and 4 minutes to read out the data. The Resource Usage tool estimates how much of the limited spacecraft and ground system resources an individual proposal consumes. This resource information is used by the Time Allocation Committee (TAC) to prevent the oversubscription of the available resources, e.g., total spacecraft time, data volume, etc.

The tool is written largely in OPS5 and uses the same set of rules used by Transformation to assemble exposures into the scheduling aggregates, as described in Jackson (1987). The overhead times for Earth occultations, slews, and guide star acquisitions depend on the number and durations of these scheduling aggregates. The earth occultation overhead times are calculated assuming a conservative estimate of the average viewing time. The Resource Usage tool's

overhead time estimates could not be significantly improved upon without actually scheduling the proposal's exposures.

4.3.3 Feasibility

The purpose of the Feasibility tools is to identify problems with exposures before the exposures are scheduled or executed. By identifying these problems early, the proposer and STScI staff have more time to devise problem-free exposures which meet the proposer's scientific needs.

Currently there is a tool to check that exposures are not requested at times which conflict with the sun distance or moon distance limits, and a tool to verify that uncalibrated filters or entrance apertures are not used. The next tools to be implemented will:

- Determine if and when guide stars are available for an exposure;

- Check that the proposer has not requested exposures which are syntactically legal but which are inconsistent, absurd, or missing crucial related exposures;

- Verify that the exposure time is consistent with the instrument used, the target brightness, and the signal to noise ratio and verify that the instrument will not be damaged by an overexposure.

The list of possible Feasibility tools is almost endless.

4.4 Selection

The Selection function is provided by a set of tools which are used to assign proposals to referees and to support the decision making process of the TAC (Time Allocation Committee) and the STScI Director on which proposals to select. The most important tool is called TACOS, a natural language interface to a single table data base and described in Hornick, Cohen, and Miller (1987). The TACOS user can create querying or editing commands either in real time or by procedures stored in the user's initialization file. A potential query might be the average referee score of all Solar System proposals. Potential data editing could be modifying referee scores or entering the TAC priority of a single proposal.

TACOS is used by the TAC to track the resources allocated and balance the accepted program between the various subdisciplines and proposers. For example, the TAC must make sure that European Space Agency member nation proposers receive 15% of the HST observing time and that real time spacecraft contacts

The TACOS tool is written in Common LISP and can be used on any single table database. The syntax, grammar, and data base structure are all determine by initialization files and the tool could be used on a completely non-HST problem. Once the initialization files are created, the users can create their own commands and procedures with this meta-tool.

4.5 Transformation

The Transformation Subsystem, described in Rosenthal, Monger, Miller, and Johnston (1986), converts the information in the IDB into the representation required by the ground system data base (PMDB). It is an expert system written in OPS5 and C and currently contains about 550 rules. It provides an interface between the Astronomer-friendly syntax of the Proposal Forms and the complex and voluminous syntax of the ground system.

The subsystem is designed and built in an environment where the users were developing and would continue to develop the procedures for populating the SOGS input data structures from IDB data structures in PEP. Transformation has been in operation for more than two years and is continually being enhanced to deal with additional proposal syntax and with new procedures. Converting the data using a rule based expert system has proven to be a very effective way of meeting the rapidly evolving requirements of the users.

5. Remote Proposal Submission System

An alternative method of entering proposal data into the PEP Entry Data Base is via the Remote Proposal Submission System (RPSS). The goals for RPSS were to provide an easy to use system with wide user access. The system is consistent with the paper forms and allows the users to detect and fix syntax problems with their proposals.

A standalone computer was used to provide the PEP Entry Subsystem functions to users logged in to RPSS from remote sites. The separate computer provides necessary processing power, network connections, and security provisions. Based on the successful experience with this concept, we are beginning distribution of parts of the RPSS software to remote sites for local usage.

Under the current procedures for proposal processing, the prospective HST user will send the Coverpage, General Form, and the Observation Summary Form, to STScI in the period called Phase I. The Director and TAC will select proposals, based on this information. The successful proposers (General Observers or GO's) will then use RPSS to transmit their Target Lists and Exposure Logsheets to STScI, in the period called Phase II. The following sections describe the operation of RPSS in more detail.

5.1 RPSS File Format

The RPSS representation of the proposal forms was intended to resemble the paper forms as closely as possible and to be simple to use. The RPSS remote proposal file is an ASCII flat file, with each line containing either a RPSS keyword and optional value or else a comment, i.e.,

Keyword : Value

The RPSS remote proposal file is organized in **blocks**, **records**, and **lines**. **Lines** are the smallest element are grouped into **records**. **Records** are grouped into **blocks**. **Blocks** correspond to different parts of the proposal forms. The valid **blocks** are:

Proposal Form	RPSS Block Keywords
Coverpage	coverpage
General Form	abstract general_form_proposers general_form_text general_form_address
Target List	fixed_targets generic_targets solar_system_targets
Exposure Logsheet	exposure_logsheet
Scan Data	scan_data

A record is analogous to single line on the Target List or Exposure Logsheet and is a logical collection of RPSS lines. The start of a record is signaled by the use of a record keyword. For example, the Fixed Target List paper form can contain several different targets, each with a line of data on the paper form. The fixed_targets block would contain several records, each corresponding to one target on the paper form. There will be as many records in this block as there are targets on the form.

The following diagram of an abbreviated proposal shows the Block, Record, Line structure, using the actual keywords.

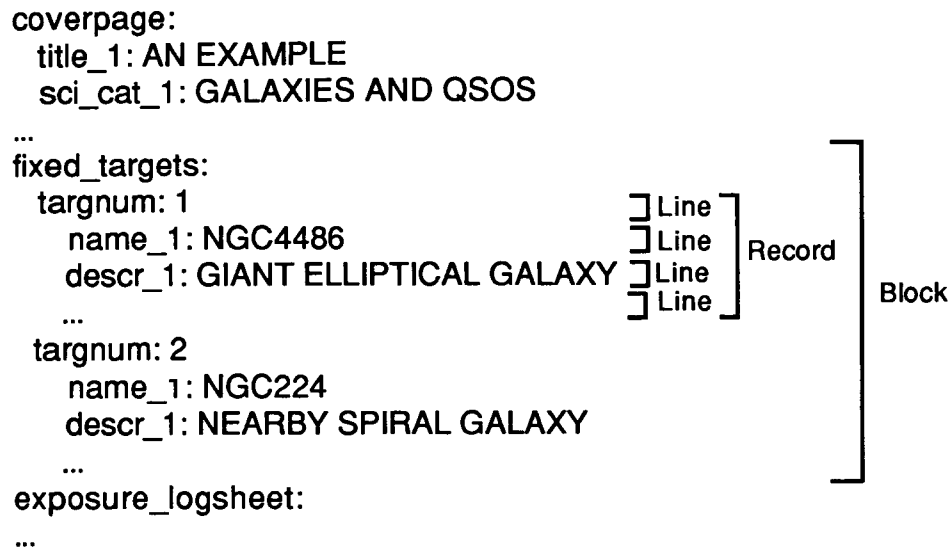


Figure 2. RPSS File Sample

A special keyword is *include*: <filename>. It is used to break a large proposal file into several smaller files. Since file transfer can be very slow, this keyword allows a small portion of a large proposal to be transferred. This is especially useful when editing on the user's computer and validating on the RPSS computer.

5.2 The RPSS Process for Phase II Observers

The successful proposers (Observer) will receive a letter from STScI announcing their selection. This letter includes instructions on how to access the RPSS node, their account name, and password. They can reach the RPSS computer via three different networks: TELENET, SPAN, and ARPANET. TELENET uses the x.25 protocol and files can be transferred using KERMIT. SPAN and ARPANET use the Decnet protocol and files can be transferred using the COPY command or VaxMail. To increase the account security, the password given to them is pre-expired, and the observers must change their password after logging into the account. If they forget, they must contact STScI to re-enable the account.

Waiting in the new account will be two VaxMail messages. The first message gives them some instructions on how to use RPSS. The second message contains the proposal information which has already been entered by STScI, i.e., the Coverpage and General Form. They can execute the VaxMail command `EXTRACT/NOHEAD <filename>` to create the RPSS remote proposal file in their RPSS directory.

The remote proposal file is normally transferred to the observer's home institution computer and edited there. Although the RPSS system supports on-line editing, the limits of current packet switched networks can make on-line editing very slow and suitable only for very small changes. The proposer edits the RPSS remote proposal file to enter all the information for the Target List and Exposure Logsheet sections. Once all the proposal data is entered into the RPSS format file, the file is transferred back to the observer's account on the RPSS computer.

The observer can also execute a command on the RPSS computer which generates a complete (albeit, empty) template file. The template file has all the valid keywords and is a blank proposal, waiting to be filled in. The observer may fill in or replicate the sections needed and delete the sections or lines that do not apply.

Once the proposal has been entered into the RPSS format file in the observer's account back on RPSS computer, it must be checked for valid RPSS file syntax and for valid proposal syntax. Invalid RPSS syntax can be misspelled keywords or improper format for a keyword's value. The RPSS syntax is checked by executing the command

CHECK <filename>

which writes an error file, syntax.err, in the observer's RPSS directory.

Invalid proposal syntax can include an invalid filter for a certain instrument or an unrecognizable Special Requirement. The proposal syntax is checked by executing the command

VALIDATE <filename>

which writes two error files, by_line.err and by_message.err, in the observer's directory and sends VaxMail to the observer's account when it is done. Because of the large CPU resources required by this function, the command adds an entry to a batch queue which allows only one process running at a time. This prevents the RPSS computer from being bogged down by simultaneous VALIDATE jobs.

If the observer has a copy of the RPSS software on a local VAX, they can perform the syntax and validation checking at their institution. Otherwise they must transmit the erroneous section of the proposal file to their home institution, edit it, transmit back to the RPSS computer, and rerun the command.

Once all errors are corrected the observer must use the SUBMIT command on RPS. For those observers with their own copy of the RPSS software, they will have to transmit the proposal to the RPSS computer and then use SUBMIT. This command runs both the CHECK and VALIDATE commands and will not accept a proposal with any CHECK or VALIDATE errors. This requirement forces the observer to create a Validation error-free proposal following the Proposal Instructions syntax.

If SUBMIT accepts the proposal, it will concatenate the proposal into one file, and place that file in a secure area. SUBMIT will also notify the proposer by sending a VaxMail message, which includes the remote proposal ID. The observer will need this ID number when sending the signed copied of the coverage to STScI.

When STScI receives the signed coverage with remote ID, they will notify the PEP Data Base Administrator (DBA). The DBA then fetches to remote proposal from the secure area and loads it into the PEP EDB. As part of the load procedure, the remote file is compared to what was entered previously into the EDB. The proposal title and principle investigator's name must match before the file will be loaded.

After the proposal has been loaded, a VaxMail message will be sent to the observer's RPSS account. By logging into their account regularly, the observer will know when the proposal has been loaded.

5.3 Security

Since the proposals are on a computer with public access, security of the data is a key feature of the RPSS design. The proposals are kept secure by using very tight file protections and limiting the commands a proposer may execute. RPSS users are confined to their own directories, and no file can ever be made publicly readable. As a further protection of the system, only a few VMS commands are permitted, and, of those permitted, the power of the commands has been greatly reduced.

For SUBMIT to work, it needs special privileges to write the proposal file to the secure area. This is not a privilege that a RPSS user should have, so SUBMIT is split into two parts. The first part is activated when the user types the SUBMIT command. This part "wakes-up" the second part and tells it which file to process. The user has no access to the second part, and the first part is smart enough to preserve the security of the system.

The process of storing the proposal file in a secure area, converting it to data base commands, and loading the proposal in the data base is all outside the control of the user. While RPSS allows proposals to be entered in the PEP EDB, the user has only one-way, one-time, single-proposal access to the EDB. It is not possible for a RPSS user to corrupt large portions of the PEP data base.

5.4 RPSS Hardware

The RPSS node is a MicroVax2 with 7 megabytes of memory. It has two 75 megabyte disk drives. Additional memory and disks are being purchased in order to reduce execution time for the Validate function and to give the users larger disk quotas.

5.5 Advantages for the RPSS User

It may seem like much more work for the GO to use RPSS instead of just sending in his proposal on paper forms. However, there are advantages for the GO using RPSS. First, there is the accuracy of the proposal. With RPSS, the GO knows that the proposal the Institute has in its data base is an exact match to what the GO entered. There is no risk of entry errors or of unreadable entries on the forms. Target coordinates are a prime example of where entry errors might have disastrous results.

Another advantage is fast turn-around on errors. A GO using RPSS can find and fix problems with the proposal quickly. A GO relying on paper forms must wait for STScI to enter, validate, evaluate errors, contact the GO, and make corrections. If the error correction process drags on too long, then the proposal may miss scheduling opportunities.

When changes to the proposal are needed, the GO with a RPSS format copy of the proposal can make the quickly changes and re-submit the proposal via RPSS.

With RPSS, the person who knows the most about the scientific requirements, the GO, is the same person who finds the legal syntax to express those requirements. The scientific staff at STScI can only make an educated guess about what the proposer wants, and sometimes those wants can not be easily communicated via letter or phone.

5.6 Advantages for STScI

STScI realizes several benefits from having GOs use RPSS. The Institute will have less staff time spent on proposal entry and correcting syntax and validation errors. This allows more time to be spent on evaluating and scheduling the proposals. Less computer resources, I/O, CPU, and Memory, will be needed on the main STScI computers if most of the proposals received have been validated on RPSS, and can be automatically loaded into the PEP data base.

With the entire process of proposal entry and validation being done more quickly, less time will elapse between when the GO writes the proposal and when the GO receives their HST data. Even for Astronomy with its billion year time scales, fast turn around is important.

6. Telescience and Applications to Other Observatories

In the area of proposal processing, the next generation of space based and ground based astronomical instruments have needs quite similar to those of HST. The sophistication of the NASA Great Observatories (e.g., AXAF) and of the European VLT require more than a cursory reading of a user manual. To adequately exploit the time variation in the capabilities of the new facilities, it may be necessary to have the observer completely specify the observations needed, which will be executed when conditions are best.

It may no longer be sensible to grant a user a fixed block of time and have the user directly control the facility in real time. The HST experience with proposal forms, syntax, remote entry, and validation are all directly relevant where the user does not make most of the telescope operation decisions in real time.

Telescience has been described as providing direct, iterative, and distributed user access to the remote device. Clearly Telescience is not possible for the user directly controlling HST. However, some of the software systems which take the proposer's information and convert the it into spacecraft commands can have Telescience aspects.

With the development of RPSS and with the distribution of RPSS software to remote user sites, the PEP Entry Subsystem now operates as a Telescience environment. RPSS allows the HST user to Enter, Edit, Validate, and Report on their proposal. RPSS allows the person most familiar with the scientific requirements to create a set of spacecraft activities to be scheduled in SOGS/SPSS.

Additional software tools to aid the user in preparing a HST proposal could also be added to the RPSS system. Such a tool might aid the user in selecting the configuration, mode, filter or grating, and optional parameters. This tool would be akin to the AI system, considered in Section 2, which would take the data requirements and determine the spacecraft activities. Another tool might combine the user's information about the target brightness with the choice of instrument and calculate an exposure time for the desired signal to noise ratio. In fact, prototypes of both of these tools have been developed at the Space Telescope European Coordinating Facility (ECF).

What of the Evaluation Subsystem functions? Can these be made more direct, interactive, and distributed? A Duplication function might be provided as a part of an HST Archive query system. The PEP Duplication tool could be enhanced to take a RPSS file and check for duplications against the archives. However, a RPSS version could not check for duplications against other GO proposals. Such a capability would require GO access to other GO proposal data, and this would violate the data privacy requirements.

The Resource Usage tool should be distributed if the users are to verify that their proposals meet any TAC imposed resource constraints. However, there would be a danger than the user would attempt to fine tune the proposal to reduce the resource usage and due to extra timing restrictions

make the proposal much harder to schedule. The Resource Usage estimate is made without creating an actual schedule and without knowledge of the actual orbital events. The orbital events seldom match the statistical assumptions of the tool, and any attempt at micro-scheduling are doomed to failure.

The Feasibility tools are the ones most suited to being made available to the user and in a distributed fashion. These tools would help the user to identify inconsistent or impossible to schedule observations at an early enough stage to allow modification by the person who knows the scientific needs best. The sooner these tools are added to RPSS, the sooner the proposer will be able to know that they have asked for legal AND feasible spacecraft activities. There will always be feasibility problems which only appear when actually scheduling the activities, but the more problems which are caught earlier, the better.

Portions of the proposal Selection function could be converted to a Telescience environment. This would eliminate the need for mailing proposals to referees and even for the Telescope Allocation Committee (TAC) to meet in the same room when selecting proposals. Once the proposals were in machine readable form, the distribution to referees and the collection of referee scores could all be handled electronically. A non-Telescience but still useful software tool would aid the TAC meeting by identifying scenarios where accepting proposals with the highest referee rankings violates resource limits or other constraints. The actual TAC meeting itself is not so easily replaced by distributed users communicating electronically. Existing technology does not provide the flexibility and high bandwidth which can sometimes be achieved in a face to face meeting.

7. Conclusion

The Proposer Entry Processor (PEP) system at STScI is a set of software tools which support the Entry, Evaluation, Selection, and Transformation of HST Proposals. With the addition of the Remote Proposal Submission System (RPSS), the PEP Entry Subsystem can now operate as a Telescience environment and provides the HST user with a faster and more responsive method of specifying exposures to be executed by the Hubble Space Telescope. Adding feasibility and proposal preparation tools to RPSS will further aid the user in creating efficient and scientifically productive HST proposals. Telescience operations of PEP can increase the system's usefulness while reducing its operating costs. Telescience can do more than just save trees.

Acknowledgements

The following people, while not authors of this paper, were instrumental in the development of the PEP system: William Cohen, Marc Damashek, Tom Hornick, Steve Lubow, Don Rosenthal, Steve Shore, Lyle Sutton.

References

- Call for Proposals and Proposal Instructions*, 1985, Space Telescope Science Institute, STScI SC-02.
- Hornick, T., Cohen, W. and Miller, G. 1987, "A Natural Language Query System for Hubble Space Telescope Proposal Selection" in *Proceedings of the 1987 Goddard Conference on Space Applications of Artificial Intelligence and Robotics*.
- Jackson, R. 1987, "Expert Systems Build By The 'Expert': An Evaluation of OPS5" in *Proceedings of the 1986 Goddard Conference on Space Applications of Artificial Intelligence and Robotics*.

Johnston, M. 1988a, "Automated Telescope Scheduling" in *Coordination of Observation Projects*, Cambridge University Press.

Johnston, M. 1988b, "Automated Observation Scheduling for the VLT" in preparation.

Miller, G., Rosenthal, D., Cohen, W. and Johnston, M. 1987, "Expert System Tools for Hubble Space Telescope Observation Scheduling" in *Proceedings of the 1987 Goddard Conference on Space Applications of Artificial Intelligence and Robotics*.

Rosenthal, D., Monger, P., Miler, G. and Johnston, M. 1986, "An Expert System for Hubble Space Telescope Ground Support" in *Proceedings of the 1986 Goddard Conference on Space Applications of Artificial Intelligence and Robotics*.

ST Proposal Entry Processor System Requirements, 1987, Space Telescope Science Institute.

Candidate Functions for Advanced Technology Implementation
in the Columbus Mission Planning Environment

Audrey Loomis
Senior Computer Scientist
Computer Sciences Corporation
4600 Powder Mill Road
Beltsville, Maryland 20705

Albrecht Kellner
Manager, Computer Science Basic Research
MBB-Erno
P.O. Box 105909
Huenefeldstrasse 1-5
D-2800 Bremen 1
West Germany

Abstract

The Columbus project is the European Space Agency's contribution to the International Space Station program. Columbus is planned to consist of three elements - a laboratory module attached to the Space Station base, a man-tended freeflyer coorbiting with the Space Station base, and a platform in polar orbit. System definition and requirements analysis for Columbus are underway, scheduled for completion in mid-1990.

This paper gives an overview of the Columbus mission planning environment and operations concept as currently defined, and identifies some of the challenges presented to software maintainers and ground segment personnel during mission operations.

The use of advanced technologies in system implementation is being explored by the authors. Both advantages of such solutions and potential problems they present are discussed, and the next steps to be taken by Columbus before targeting any functions for advanced technology implementation are summarized.

Several functions in the mission planning process have been identified as candidates for advanced technology implementation. These range from expert interaction with Columbus' data bases through activity scheduling and near-real-time response to departures from the planned timeline. Each function is described, and its potential for advanced technology implementation briefly assessed.

Candidate Functions for Advanced Technology Implementation in the Columbus Mission Planning Environment

Outline

- 1.0 The Columbus Context
 - 1.1 The Columbus Program
 - 1.2 Columbus Mission Planning Concept
- 2.0 Assessment of Artificial Intelligence (AI) for Columbus Mission Planning
 - 2.1 Challenges for Columbus Implementation
 - 2.2 Suitability of AI Solutions
 - 2.3 Requirements on AI Technology
 - 2.4 Next Steps for the Columbus Program
- 3.0 Candidate Functions for AI Implementation
 - 3.1 Data Base Interactions
 - 3.2 Scheduling Functions
 - 3.3 Support Functions
 - 3.4 Rescheduling Functions
- 4.0 Summary

1.0 The Columbus Context

This section provides a brief introduction to Columbus and its current mission planning concept.

1.1 The Columbus Program

The European Space Agency (ESA) is a partner with the United States in the International Space Station program. ESA's contribution to the International Space Station is Columbus, currently the highest priority project in the European space program. Columbus is composed of four elements:

- o The Attached Pressurized Module (APM), a laboratory module permanently attached to the Space Station infrastructure;
- o The Man-Tended Freeflyer (MTFF), a coorbiting platform intended to operate unmanned, but be man-visited for periodic reconfiguration, servicing, and sample recovery;
- o The Polar Platform (PPF), a platform in Sun-synchronous polar orbit.

Columbus is currently completing concept and architecture definition, and is about to enter a two-year detailed definition phase that will be followed by system implementation.

MBB-Erno is ESA's Prime contractor for the flight segment, with system engineering responsibilities in areas affecting all of Columbus' elements. CSC participated in defining the Columbus Phase B2 ground mission planning concept as a contractor to the Deutsche Forschungs- und Versuchsanstalt fuer Luft- und Raumfahrt (DFVLR), and is currently a subcontractor to MBB-Erno, supporting Columbus definition work.

1.2 Columbus Mission Planning Concept

The current Columbus mission planning operations concept shows three levels of planning - strategic, several mission periods in advance of a mission; tactical, one mission period in advance of a mission; and operations, during a mission. Strategic and tactical planning establish a mission's payload complement and resource baseline, and are primarily the responsibility of the European Mission Control Center (EMCC), an ESA facility. For APM operations, final planning is a U.S. responsibility.

Operations planning subdivides resource allocations from the tactical plan by User Support Operations Center (USOC) and then by user within each USOC, reserving resources required by the spacecraft/system. Each user plans his activities and submits his plan to his USOC. The USOC generates a composite plan and passes it to the POCC. The POCC prepares a composite of all USOC plans, which is merged with the spacecraft plan prepared by the Mission Control Center (MCC) and foreign POCC plans, if applicable. At this time, it appears likely that each element's MCC will be colocated with its POCC. Final plans are nominally one week long. A high degree of commonality in the planning software for an element, across planning levels and especially across locations involved in planning, is desirable, but may not be obtainable.

Columbus expects to support telescience. Users may plan to the command level or may not produce a refined plan at all, retaining their operations envelopes as assigned. Flight software is expected to be able to generate commands from a high-level, mnemonic-and-parameter form called an action.

A basic data structure supporting this process is the action description. Each action resembles a goal statement, and is hierarchically decomposed into successively lower-level goals until the level of individual onboard subsystem activities is reached. Each lowest-level action maps onto a group of spacecraft or instrument commands called an automated procedure. Each level of action is accompanied by information describing timing flexibility, resource profiles, and constraints. Coarse planning is done using a high-level action; detailed planning accesses the lowest-level actions.

2.0 Assessment of Artificial Intelligence (AI) for Columbus Mission Planning

This section presents the challenges that make AI solutions look attractive for Columbus and summarizes the advantages and disadvantages of using AI. It also outlines additional work that must be accomplished before Columbus can actually target specific mission planning applications for AI implementation.

2.1 Challenges for Columbus Implementation

Columbus implementation presents special challenges in the following areas.

- o Mission philosophy. Columbus, in common with the U.S. Space Station program, plans to emphasize decentralized planning, scheduling, and command generation, and to support telepresence. In order to provide effective support, software must be modular, very flexible, and possess a high degree of commonality across facilities. Support for telepresence will require innovative software designs and as much automation as possible.
- o Spacecraft and experiment support. The International Space Station era will see continued growth in the variety and complexity of spacecraft and instrument hardware. Columbus must support spacecraft designed to provide services to a wide range of experiments and to instruments with different degrees of intelligence. Platforms, the Space Station base, and instruments will certainly evolve during the International Space Station mission, and support software must evolve with them.
- o Maintenance. With payload changeovers up to every 90 days, Columbus will be driven by the need for efficient software updates. Software and data bases must be developed, tested, integrated, and made operational faster than ever before.
- o Operations. The cost of operations support during the lifetime of Columbus may be a significant portion of the cost of the program, and must be carefully controlled. The cost of operations can be reduced through the use of a highly automated, user-friendly ground support system.

These challenges apply across the Columbus mission. Some specific considerations are mentioned in Section 3.

In order to meet its challenges, Columbus will need flexible software to perform functions that are commonly either manual or non-interactive and relatively inflexible. Some of these functions are currently manual because attempts to automate them have not been successful. Some currently automated functions

require innovative implementations in order to perform acceptably in the Columbus environment.

2.2 Suitability of AI Solutions

Before discussing the advantages of AI solutions for Columbus, we present our definition of AI. We wish to avoid philosophical discussions relating to the amount of "intelligence" in AI software; our definition is based on the information processing characteristics of the system. An AI system is characterized by

- o Use of knowledge represented symbolically, such as by facts, relations, and structured abstract knowledge, in addition to numeric data;
- o Generalized information processing methods, such as feature recognition, pattern matching, and data driven search; and
- o Separation of knowledge (and data) from reasoning mechanisms.

Well-applied AI technology in the Columbus mission planning context will provide benefits in several areas. Such systems should be capable of handling problems of greater complexity than a human operator can handle within an acceptable time frame. The inference engine/knowledge base structure eases the software maintenance problem by allowing the software implementing strategies and approaches to remain unchanged when the environment upon which it operates changes. The cost-effectiveness of operations will increase with increased automation, as long as the software is user-friendly. AI technology has the potential to explain the software's processing to the user, simplify data presentation by locating the data with the most significant information content, and provide associated information based on an understanding of the user's intent.

Benefits of applying AI technology to specific functions are given in Section 3.

2.3 Requirements on AI Technology

The use of AI systems as mainstream, operational software will undoubtedly present technological challenges in the areas of knowledge representation and reasoning. Such research problems are already receiving attention, and more focussed work will be done as the International Space Station requirements develop.

The integration of AI software into a complex operations infrastructure; its interface with numerous program elements and data sources; and the need to generate, verify, and maintain large knowledge bases, raise challenges in the management area. These problems are receiving less attention, and we identify two here.

2.3.1 AI software development and management

Before deciding on AI developments, it must be possible to plan a cost-effective development effort. Operational software must work correctly and on schedule.

A suitable development environment, tailored to the types of AI implementations being targeted, is required.

A commitment to early prototyping is also required. The feasibility of each application must be proved, especially in those areas in which the technology itself must be stretched. We cannot assign a difficult job to an expert system, only to find out during development that the desired system is not feasible. Also, sizing estimates and productivity estimates are not easy to make for AI software; too little commercial history is available. Construction of prototype software will help to scope the difficulty of the job so that the full-scale implementation can be scheduled correctly.

Finally, AI development should be guided by the use of a development methodology just as traditional software development is. We find a comprehensive development methodology is necessary to monitor and control the implementation of traditional systems. Less well-understood AI implementations are at least equally in need of monitoring and controlling.

2.3.2 Knowledge management

In order to be effective, AI systems for spacecraft support must be able to access and process extremely large amounts of data and knowledge. For example, an AI-based mission planning and scheduling system would require descriptions of a potentially significant number of operations envelopes (actions) for spacecraft and payload activities. Each envelope is a large data structure. Roughly twenty additional distinct types of input have been identified.

The effort and cost involved in generating, verifying, updating, and maintaining such large knowledge bases could be great. Large portions of the required information are already being provided by the ground infrastructure; they are part of the mission data bases. However, the ability to extract and reformat the information for use by AI software is not currently available. If data base information could be transformed into knowledge bases, or used as such, the problem of knowledge management could be relegated to a large extent to the development, verification, and maintenance of data bases which are already an integral part of the ground system. This would eliminate the need for parallel development, verification, and maintenance of knowledge bases.

2.4 Next Steps for the Columbus Program

We have begun to consider AI applications for Columbus in the early stages of system definition, because we realize that, for such applications to be feasible, they must be planned for. However, before any specific mission planning applications can be targeted for AI implementation, several analysis steps must be performed by the Columbus program.

First, a more detailed understanding of the requirements placed upon the Columbus mission planning system is necessary. No implementation can be effectively designed until the requirements are complete and consistent.

Policy decisions about the mission planning system architecture are also required. Mission planning functions are needed in a number of different physical locations and by a number of different types of users. If common mission planning software is not used, the various planning systems must be built to very restrictive specifications, so that, as timelines progress from the users to the USOCs to the POCs, they are treated consistently. Opportunities for options such as expert system support for scheduling might be limited, even if they are technically indicated. If common software is used, the use of AI techniques should be much more possible.

Finally, the management challenges mentioned in Section 2.3 must be addressed.

3.0 Candidate Functions for AI Implementation

This section describes mission planning-related functions that are candidates for AI implementations and briefly assesses the potential of each implementation.

3.1 Data Base Interactions

Data base interactions are those functions related to definition or retrieval of data base information.

3.1.1 Action definition/validation

Action definition is the task of populating the action portion of the mission planning data base. Action validation is the task of determining the consistency (and, if possible, the completeness and correctness) of the defined actions. Action definition and validation software should execute either interactively, as a user defines an action (definition and validation functions), or in batch, to check pre-determined actions (validation function only).

The definition task does not require expert system implementation; however, it lends itself to hierarchical

implementations such as structured objects or search trees/graphs. If such structures are used, they automatically provide a foundation for capabilities such as inheritance between levels of actions. Checking for consistency and completeness is also simplified.

The task of determining the correctness of action definitions is traditionally a human responsibility. However, it seems to be a direct candidate for expert system implementation, where the expert system would function as an assistant to the responsible person. An action-checking expert system might be extended to check automated procedures.

These tasks can be implemented without extending the state of the art in AI, and they do not present severe performance restrictions. The risk associated with defining these functions as AI candidates is minimal.

3.1.2 Report format validation

The report format validation task is a simplified version of action validation. Columbus users will be allowed to define their own reports using SQL-like functionality. Inexperienced users are not always aware of which parameters are relevant in a particular context, and may benefit from such advice (for example, all reports dealing with stored commands should include their timetags in spacecraft clock counts as well as in GMT). The same type of software that checks the completeness of actions could check the completeness of report formats.

3.1.3 Knowledge base validation

The introduction of knowledge-based software into an operational system raises the problem of knowledge base maintenance, especially in the case of large or dynamic knowledge bases. Although modifying rules, or input logically equivalent to rules, is generally easier than modifying software, it is not necessarily straight-forward. A detailed knowledge of the inference engine that uses the data is required in order to gauge the effects of a knowledge base update. The best solution to this problem, on paper, is to construct a comprehensive set of benchmark tests including the exercise of different planning strategies and rerun them with every change of the knowledge base. Experience has revealed that this is not practical, and work is currently being done in the area of designing expert systems to help limit the scope of the problem.

Complete knowledge base changeovers also require extensive debugging to check syntax, dependencies, consistency, and structure (order, use of triggers, etc.). Different development environments provide different degrees of support for this job. Some debugging must obviously be done through test runs. If an expert system could be developed to "fill the gap" between development environment capability and functional tests, it would

be a cost-reducing contribution to system maintenance. Columbus' frequent payload changeovers will create an unprecedented need for fast, absolutely correct software system updates.

Current expert system technology is equal to the knowledge base update problem; identifying effective, efficient techniques for knowledge base interpretation is a current research topic.

We suspect that Columbus' largest knowledge bases will not belong to the planning tool, but to interfacing subsystems such as on-orbit fault diagnosis; the issue is raised here for completeness.

3.1.4 Scheduling software interaction with action data base

Columbus anticipates using a relational data base to store action definition and expansion data needed by the scheduling software. In a recent exercise, CSC modeled the scheduling of a simple instrument activity for Landsat 6 using Columbus' action concept. Although the instrument commanding was trivial, the supporting spacecraft commanding was dependent upon the context in which the instrument activity was scheduled. The definition and retrieval of these context-sensitive activities was very difficult.

One approach to solving this problem is to implement an intelligent data base access capability in the mission planning software. This function would use the "core" action and the current scheduling context to identify, retrieve, and instantiate supporting actions.

The most difficult part of this task is identifying supporting actions based upon context. This problem has been solved in the past by developing multi-pass software with fewer degrees of freedom as more passes are completed. This approach has been very successful when the scheduling of one type of activity essentially determines the rest of the schedule or when one heuristic heavily outweighs all others. In a well-balanced world, such an approach generates either inefficient schedules or inefficient software.

Columbus' scheduling drivers have not yet been identified, and the prevalence of context-sensitive activities is not known. Therefore, no specific approach to the problem can be suggested at this time. However, the instantiation of context-sensitive actions is recognized as one class of scheduling software / data base interactions requiring an innovative solution.

3.2 Scheduling Functions

Scheduling functions are those related to the definition of a new timeline or the addition of information to an existing timeline.

3.2.1 Scheduling strategies

The efficiency of the scheduling function strongly depends upon the

strategy embodied in the process of selecting requests for the generation of a constraint-free timeline. The more complex the scheduling process itself becomes, the more likely the need for heuristics in request selection becomes. A set of well-chosen heuristics may produce much better results than a simple algorithm would produce, due to the nature of the factors to be considered. Priority factors, such as scientific priority, degree to which scientific goals have been met, and geographical return, will already have been factored into the data base before execution of the scheduling software. The request selection function must consider factors such as urgency (number of remaining opportunities to schedule a request), interruptibility of the request, duration of the request, resource usage (especially usage of over-committed or difficult-to-obtain resources), and the amount of flexibility the request presents to the scheduler (size of scheduling window, specification of alternate resources), in addition to the data base priority. Factors such as these will produce conflicting selections; therefore, the use of heuristics is indicated.

The biggest challenge this implementation is likely to present is performance. There is a tradeoff between amount of intelligence and software performance, particularly as the number of requests to select between rises, or as the number of requests with nearly identical characteristics rises, depending upon the actual heuristics chosen. Request selection heuristics should be developed in tandem with scheduling heuristics, so as to provide the most effective preprocessing for the scheduler. Performance tradeoffs should consider the performance of the scheduler; if more intelligent request selection makes a significant difference in scheduling performance, then the extra execution time is worthwhile.

3.2.2 Realization aspects

Automatic spacecraft scheduling employing AI techniques is a common research topic at present. We have seen several small systems, some of which are operationally useful, but none that can handle a problem of Columbus' magnitude. Examination of systems under development presents a variety of ideas to guide the definition of a system for Columbus.

Performance requirements dictate that the scheduling system should not be more general than necessary. When Columbus' problem is more fully understood, it will be possible to further limit the scope of the implementation. Issues to be investigated include

- balance (or lack of it) in resource oversubscription (is one particular resource a bottleneck?)
- percent of unconstrained time available for an activity type or experiment (how easy is it to locate potential time slots?)

- percent of input that can be scheduled (will almost all of a set of requests fit onto a timeline, or only a small part?)

Answers to these questions and related issues will help to define effective heuristics for manipulating timing variability associated with an activity and selecting a good placement on the timeline.

Even within the Columbus environment, overgeneralization is possible. Desirable scheduling heuristics for crew activities may differ from heuristics for experiment scheduling. (A heuristic that schedules the minimum duration for an activity and then expands it may be valuable for crew activities, because it would provide a time cushion for the performance of an activity. The same heuristic, applied to experiments, would tend to allow the maximum amount of time to warm up an instrument and the minimum amount of time to use it.)

System performance may still be a concern, even after avoiding too general a solution. The development hardware and software weaknesses must be identified and avoided. Standard problems include various memory limitations and performance elbows related to system characteristics such as depth of nesting of inheriting objects, number of rules, number of variables in a pattern-match, and number of LHS clauses. A modular system design can overcome many of these problems and also allow for easier expansion. Options include use of objects, particularly generic ones; use of triggers and demons, so that some context is defined when an action takes place; knowledge base partitioning; and use of a set of cooperating expert systems, each concerned with a single aspect of the problem. The most significant key to performance lies in the right choice of scheduling heuristics, strategies, and methods of search space reduction. Heuristics designed to diagnose scheduling failures may be helpful. Current scheduling work shows that scheduling by use of time zones with constant resource usage and constraint characteristics is more efficient than using time zones of constant duration.

Finally, such a system must not be under-automated or linked together with manual processes. Current systems originally designed as tools are evolving toward full automation and batch execution. An operator probably cannot materially improve a complex, constrained schedule generated by software, and cannot afford to be tied to the keyboard attempting to construct one manually.

Columbus will push the limits of current technology, not so much in concept as in the sheer size of the problem. Detailed analysis of potential implementation techniques and scheduling heuristics will undoubtedly surface questions requiring additional research to answer. The problem of knowledge base implementation will be non-trivial due to the vast amount of data required. Unlike traditional software, AI does not have a rich

commercial history from which to draw lessons on the use and maintenance of a large, long-lived, operational system.

The primary benefits of an automatic scheduling implementation are expected to be an increased probability of obtaining a satisfactory schedule in a reasonable amount of time, greatly increased maintainability, and, if modular knowledge organization is used, universality; an experimenter can use only those portions of the knowledge base that apply to him, while the entire knowledge base is available to the POCC/MCC.

3.3 Support Functions

Support functions are part of the scheduling subsystem, but not part of the scheduler itself.

3.3.1 Priority adjustment

The Columbus Flight Operations Office (FOO) wishes the operational mission planning system to be able to recommend alterations to user/experiment priorities during a mission. Priority adjustment software would use planned timelines (or, possibly, as-flown timelines) as input, and make recommendations based upon criteria such as geographical return, degree to which science goals were met, relative priority of the particular science, and political considerations. An expert system with a good explanation facility would be a natural format for this software.

3.3.2 Command and command parameter generation

The task of deriving a detailed description of support activities, command sequences, and command parameters can be very complex. For example, while it may be sufficient to indicate the start and end of a TDRSS downlink on a timeline, an operational description of the desired contact currently requires up to 56 parameters, which must be internally consistent. As system automation increases, the demand for this type of processing is expected to increase. This function is regarded as a candidate for AI implementation, but cannot be analyzed further at this time.

3.3.3 Command checking

A great deal of command checking is traditionally automated at various points during command generation, command load generation, and load uplink. The incorporation of telescience produces an inability to assemble the total commanding picture for an element during the preplanning time frame, and demands a new approach to the current command checking process. Although the core of the solution to this problem is the development of an innovative operations concept, command checking may directly benefit from AI technology by an expansion of the scope of the automated checks. Semantic checks have not traditionally been

automated. (Semantic checks are those which validate the selection of a particular command or the composition of a procedure, through understanding the meaning of the command(s).) Pre- and post-condition checks, on the command level, are closely related to semantic checks. If this type of validation were automated, some of the risk incurred by permitting separate command stream uplinks would be alleviated.

3.4 Rescheduling Functions

Rescheduling functions are those related to the modification of an existing timeline due to a change in the baseline under which it was created.

3.4.1 Replanning strategies (reintercepting a timeline)

The task of reintercepting a timeline following recovery from a deviation from planned activities has generally been handled in one of two ways - either manually or by replanning the entire timeline in software, effectively ignoring the problem. Reinterception is defined as bringing actual conditions to match planned conditions at some point of time, after which the planned timeline may be reactivated. Intelligent replanning strategies need to select a reintercept point (which might be the end of the planned timeline), recognize what to do in order to bring about convergence, factor in the amount of time between the start of the replan and the time of convergence, and generate all required activities. Sample heuristics include minimizing time to convergence and minimizing the number of affected activities.

Automated replanning is a very difficult task. The complexity of the job rises as the number of conditions that must be manipulated rises, or as the efficiency requirements placed upon the new plan become more stringent. The number of conditions to be manipulated determines the difficulty of locating a convergence point, and of generating actions to promote convergence. If a near-optimal replan is required, then comparatively easy solutions such as dropping activities from the schedule until remaining activities are constraint-free are not acceptable.

Performance, data availability, and heuristic limitations combine to make any implementation difficult, and Columbus' problem is not well-bounded. As Columbus' requirements develop, the following analyses should be made:

- How many conditions are important to convergence? (How hard is the replanning problem?)
- How capable is the onboard safing system? (What will cause a need for replanning?)
- What do the planning heuristics look like? How closely can replanning tie into the initial planning capability? (What is the cost/risk of implementing automated replanning? There is a trade-off between acceptable cost/risk and the amount of real need for sophisticated replanning.)

Test-bed software will be very useful for analyzing the problem of identifying information required by replanning software and for experimenting with heuristics once a semi-realistic data environment has been generated.

3.4.2 Command execution monitoring

Columbus, and other International Space Station era projects, will need a far more sophisticated capability for command execution monitoring than has yet been implemented. Telescience will cause a large impact on traditional system designs, which typically assume that at some point in time, command streams can be assembled and checked. In the future, the first chance to examine the results of issuing a command may well be during real-time execution monitoring.

An ideal system would have a highly advanced capability to monitor command execution, detect any undesirable consequences early and identify the cause, decide what to do (safe the spacecraft? safe the payload? request replanning?), and take initial steps to maintain spacecraft health and safety and as much science as possible. Some of the primary considerations for successful implementation are:

- provision of sufficiently inclusive monitoring points, so that command execution can be effectively monitored
- implementation of a system-wide approach to safing that allows options to be unambiguously defined, and that describes criteria for selecting between options
- recognition of the limitations of onboard computer capacity, both storage and processing speed

In this context, we are interested in command execution monitoring because of its potential for requesting replanning. Eventually, it would be desirable to see onboard software that could determine the minimum shut-down necessary to maintain health and safety and generate some replanning parameters such as how soon a new timeline must be received and characteristics of the new timeline (e.g., 20% less power, no SSA downlinks).

Performance and reliability are two of the many challenges this problem presents. Performance will almost certainly be a problem, due to the need to keep pace with input from monitors, the extensive processing required to analyze such input, and the occasionally complex nature of the required analysis. Telemetry input to expert systems is frequently handled by using a separate front-end processor, sometimes running on special hardware, to subset and reformat telemetry points at selected intervals. This solution is not promising for onboard systems. The amount of processing attempted by a monitoring system can be scaled to given performance limits, but in so doing the system may become trivial. The reliability problem is two-fold. First, an active system must not make mistakes; it must execute consistently and reliably. Second, the system must be perceived as reliable. People who are responsible for the health and safety of onboard

crew, or of expensive spacecraft, must be persuaded to allow an active command execution monitoring system to handle complex safing and request replanning. This may be as great a barrier as any technology issue.

4.0 Summary

This paper introduced the Columbus mission planning environment and described key advantages and problems associated with implementing parts of the mission planning software as AI systems. Functions identified as potential AI candidates include several types of data base interactions, scheduling functions, support software, and rescheduling functions. Before actually targeting any of these for implementation using AI techniques, Columbus' concepts and requirements must be precisely defined, and research into solutions to the problems of developing usable, large-scale, mainstream AI systems must be undertaken.

A Rule-Based Systems Approach to Spacecraft Communications Configuration Optimization

James L. Rash, Yen F. Wong, and James J. Cieplak

Telecommunication Systems Branch, Code 531
NASA Goddard Space Flight Center
Greenbelt, Maryland

Abstract -- An experimental rule-based system for optimizing user spacecraft communications configurations has been developed at NASA to support mission planning for spacecraft that obtain telecommunications services through NASA's Tracking and Data Relay Satellite System. Designated ECCO (Expert for Communications Configuration Optimization), and implemented in the OPS5 production system language, the system has shown the validity of a rule-based systems approach to this optimization problem. This paper discusses the development of ECCO and the incremental optimization method on which it is based. A test case using hypothetical mission data is included to demonstrate the optimization concept.

I. INTRODUCTION

Spacecraft in low earth orbit may obtain tracking and communications services through NASA's Tracking and Data Relay Satellite System (TDRSS), provided they are TDRSS compatible [1]. Such user spacecraft span a wide range of mission types, including scientific satellites and planetary probes, Space Shuttle, and Space Station.

Complexities of mission requirements and the wide variety of TDRSS users call for a tool to analyze the communications performance, and TDRSS compatibility, of user spacecraft. Such a tool, the Communications Link Analysis and Simulation System (CLASS), has been developed by the NASA Goddard Space Flight Center in Greenbelt, Maryland [2]. CLASS performs its function by end-to-end modeling of all elements of the communications link.

Within CLASS, the Flight Performance System (FPS) is a software capability for predicting performance of TDRSS-supported missions under simulated flight conditions, orbital or nonorbital. FPS is designed to generate all required mission and communications performance data and channel condition indicators, both in real time and in a mission planning mode.

FPS is now being given the capability to find the user spacecraft communications configuration that will provide optimum communications performance under given constraints in a computer simulation environment. This paper presents ECCO (Expert for Communications Configuration Optimization), a rule-based system providing this capability to mission planners. Section II below discusses the optimization problem and the method for its solution, while Section III discusses the design and implementation of ECCO. Section IV presents a test case illustrating the optimization concept. Finally, Section V summarizes current status and possible further development.

II. COMMUNICATIONS CONFIGURATION OPTIMIZATION

A major objective of spacecraft mission planning is assurance of reliable space communications. For some time now it has been possible for mission planners to accomplish this by means of CLASS software tools that predict and evaluate communications link performance [2]. The CLASS also can be used to perform various kinds of optimization. Of great interest to mission planners is the optimization of spacecraft communications configurations based on bit error rate (BER) performance.

A. TDRSS Telecommunications Service Overview

The Tracking and Data Relay Satellite System consists of a space segment and a ground segment as shown in Fig. 1. The ground segment of TDRSS consists of a ground terminal at White Sands, New Mexico. Ultimately, the operational space segment of TDRSS will consist of two in-service satellites in geostationary orbit at 41 and 171 degrees west longitude. By this arrangement, communications coverage of user spacecraft will be at least 85% [1]. Though not planned for use except in emergency, an additional satellite will be stationed as a spare at 61 degrees west longitude.

TDRSS provides tracking and data acquisition services to user spacecraft, with multiple access at S-band (SMA) and single access at S and Ku band (SSA and KSA).

Spacecraft utilizing TDRSS have wide-ranging communications characteristics -- differing numbers of defined communications links; antennas of different types, sizes, and number; different data coding schemes and transmission rates; different signal polarization and power levels; etc. These and all other pieces of information necessary to characterize the communication systems of TDRSS and user spacecraft, as well as the channel environments, are captured in CLASS data bases.

B. Optimization Concepts

The rule-based system discussed in this paper, ECCO, assumes that all characteristics of the user spacecraft communications system (frequency, data rate, transmitter power, etc) are compatible with TDRSS.

Given a user spacecraft and its mission constraints, and given a specific time point during the mission, the goal of communications configuration optimization is simply the selection of the best alternative from among all allowable communications configurations of the user spacecraft.

A user spacecraft *communications configuration* is defined as a set of *link combinations*. Each link combination is specified by four *controllable parameters* together with a measure of communications performance (BER margin). For the present discussion, the four controllable parameters are (a) user antenna, (b) supporting TDRS, (c) supporting TDRS antenna, and (d) user communications link in use at the mission time point in question.

ECCO uses BER margin as calculated by FPS to determine the user communications configuration (i.e., the set of link combinations) that will provide optimum communications performance.

Assuming that there exists at least one viable communications configuration of the user spacecraft at the given mission time point, ECCO performs optimization incrementally by means of successive eliminations from the set of all allowable communications configurations that exist for that time point. Each of these elimination steps is essentially a test to be applied to all of the alternative configurations in that set. ECCO is designed to apply these elimination steps in a particular order of priority.

This approach to finding the optimum communications configuration is referred to as an *incremental optimization method* and is described formally as follows:

- (a) Form the set K of all user spacecraft communications configurations that are allowable at the mission time point in question.
- (b) Incrementally apply all elimination steps, in the prescribed order, to the set K.
- (c) If the resultant set K contains more than one member (i.e., no way exists, based on the elimination rules, to discriminate between members of the set, thus identifying one as better than another), then select a member of K at random and designate it as the optimum configuration.

A set of optimization rules following this general strategy will be discussed below in Section III. E.

III. ECCO DESIGN/IMPLEMENTATION

A. ECCO Design/Implementation Overview

ECCO is a rule-based system to find an optimum communications configuration of the user spacecraft at a prescribed mission time point, if such a configuration exists. The experimental version of ECCO discussed in this paper performs *single point optimization* in which the discovered optimum configuration is independent of earlier or later events. The future *operational* version of ECCO, however, will perform multipoint optimization which will reflect the effects of earlier and later mission events. As a consequence of the single point optimization requirement, the present experimental version of ECCO must read in all necessary input relative to the given

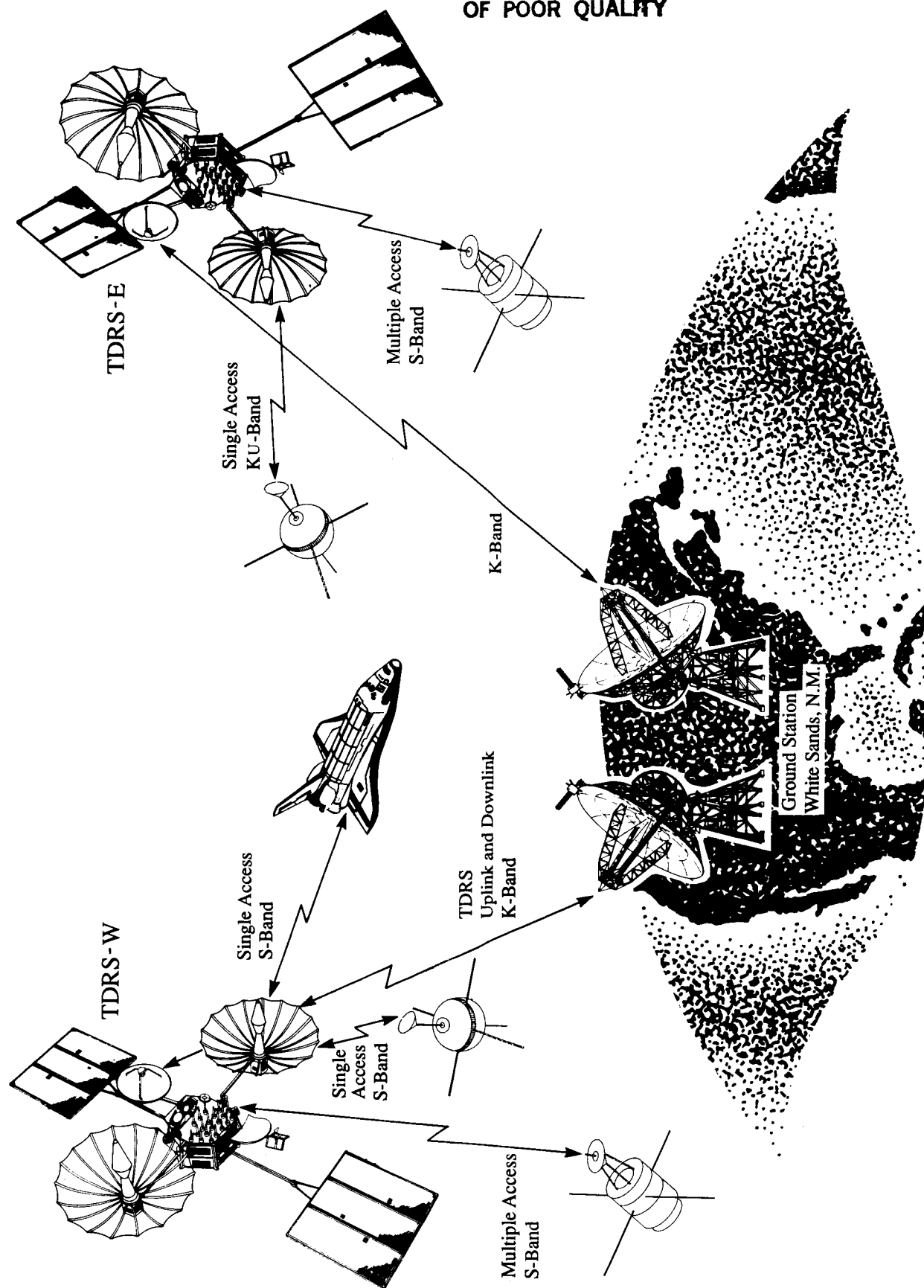


Fig. 1. TDRSS/User Configuration

mission time point prior to beginning the optimization process. The future operational version of the system ultimately must function in a completely unattended mode, performing all input and output activities automatically.

Initial capabilities of ECCO must accommodate all TDRSS-compatible spacecraft except Space Shuttle.

B. FPS/ECCO Data Interfaces

Fig. 2 depicts the data interfaces between FPS and the future operational version of ECCO. As illustrated in Fig. 2, the sources of FPS input data are (1) mission data tapes supplied by the user spacecraft project, (2) CLASS data bases, (3) data from attitude/orbit generators, and (4) real-time update terminals. FPS performs all calculations pertaining to mission and communications performance. These calculations include:

- o Signal margin (BER)
- o TDRS visibility
- o Environment data (atmospheric loss, RFI loss, sun interference, etc.)
- o Vehicle multipath/blockage
- o Probability of loss of lock

TDRS visibility and vehicle blockage calculations are needed only to determine whether a line-of-sight path exists between a given TDRS and a given user spacecraft antenna. Only signal margin (BER) is used directly by ECCO. Other parameters calculated by FPS are not considered in the present experimental version of ECCO.

C. ECCO Input

ECCO depends on FPS to calculate BER margin for each possible communications link between the user spacecraft and TDRSS. These BER margins, if acceptable (i.e., nonnegative), are contained in the *link combination table* generated by FPS. As indicated in Fig. 2., the other required inputs for ECCO are the *link definition table* and the *link substitution table* which are provided in the form of CLASS data base inputs by the user.

1. Link Combination Table

The link combination table contains the following data items:

- (1) Link combination sequence number (arbitrarily assigned by FPS)
- (2) Link ID
- (3) User antenna ID
- (4) Supporting TDRS ID
- (5) TDRS antenna ID
- (6) BER margin (dB) for I-channel if link is a return link or for entire link if it is a forward link
- (7) BER margin (dB) for Q-channel if link is a return link
- (8) Total BER margin (dB)

For forward links this is simply the forward link BER margin. For return links this is the sum of the I-channel and Q-channel BER margins.

FPS generates a link combination table for each mission time point where ECCO is enabled. To generate this table, FPS calculates the BER margin (items (6) and (7)) for all possible link combinations from the user spacecraft to each in-service TDRS (by letting the values of items (2) through (5) take on all allowable values). Each link combination found to have an acceptable (nonnegative) BER margin is included in the link combination table, whether the link ID is marked as active or not.

2. Link Definition Table

The link definition table (supplied in advance by the user) contains the following data items:

- (1) Link ID
- (2) Link type (Forward/Return)
- (3) Service category (Multiple Access/S-band Single Access/K-band Single Access)
- (4) Frequency in Hz x 100

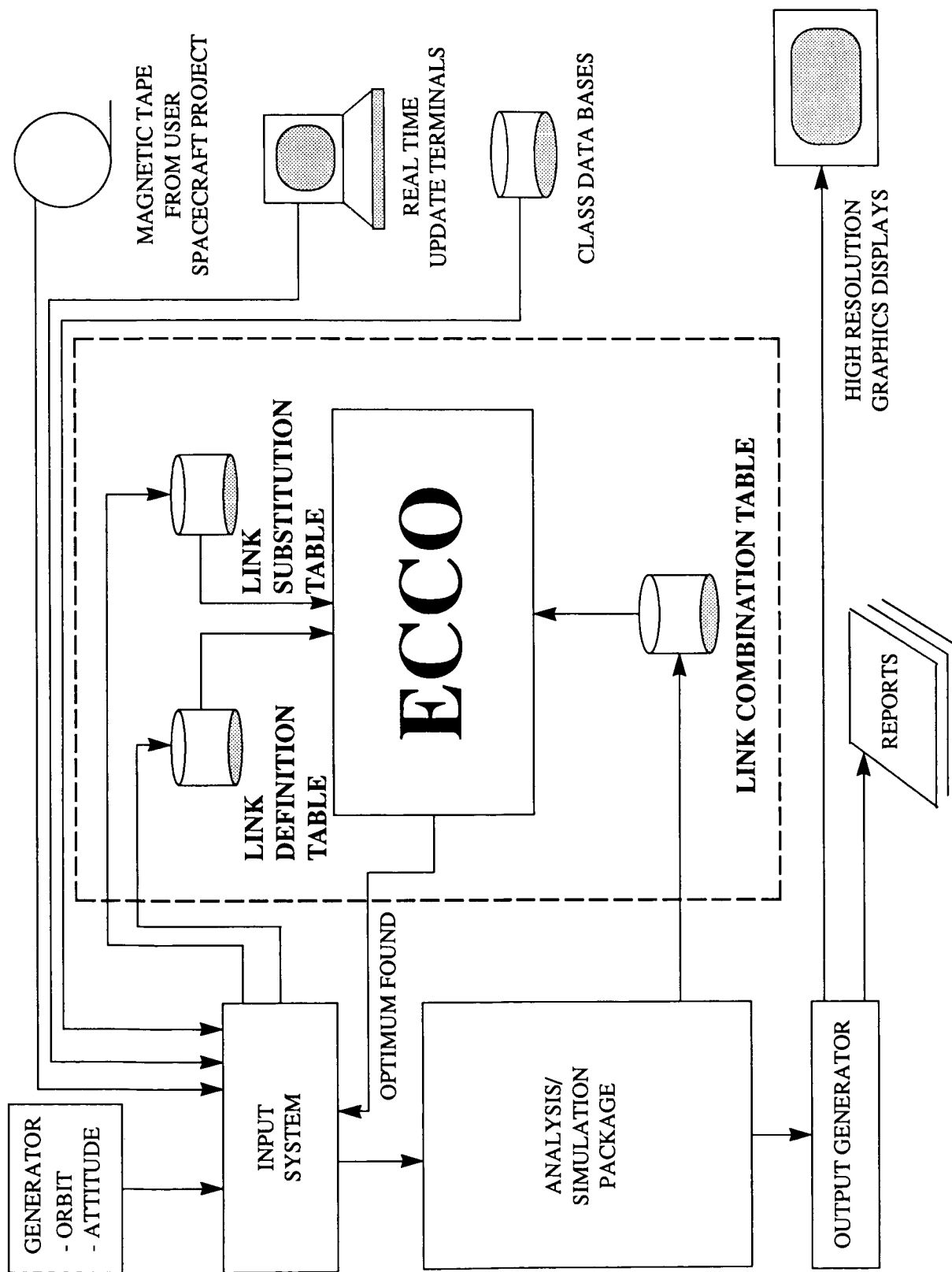


Fig. 2. CLASS/FPS/ECCO Data Interfaces

- (5) I-channel data rate (bits per second)
If link is a forward link, this is the data rate for the entire link.
- (6) Q-channel data rate (bits per second)
If link is a forward link, this is set to zero.
- (7) Data Group
- (8) Mode
The data group and mode are arbitrary parameters used as a means for classifying transmission signal characteristics (data rate, spread/unspread spectrum, and coherency/noncoherency).
- (9) Priority designation (Priority/Nonpriority)
- (10) Status (Active/Inactive)

Each link ID defined for the mission is included in this table. If a link ID is active and is designated as a priority link, it will be given highest preference in optimization.

3. Link Substitution Table

The link substitution table (supplied in advance by the user) lists ordered pairs of link IDs. The second link ID in each pair is a valid substitute for the first, as prescribed by the user spacecraft project office. Any given link may have more than one substitute, or none.

D. ECCO Output

Output from the present version of ECCO consists of the discovered optimum communications configuration of the user spacecraft for the given mission time point. It can be used by a mission planner in revising the mission plan to improve overall communications performance during the mission.

The future operational version of ECCO will have an automatic mode of operation allowing FPS to modify the mission plan itself, based on ECCO output, in order to obtain optimum communications performance for the mission as a whole.

E. Optimization Rules

The processing steps in ECCO are derived from a set of rules based on TDRSS telecommunication system design as well as the mission planning knowledge of a domain expert, and follow the general strategy referred to earlier as the incremental optimization method. These rules are as follows:

- (1) Data input.
Input the link combination table for the given mission time point. Input both the link definition table and the link substitution table for the mission.
- (2) Link substitutions.
- (3) Generation of communications configurations.
Generate the set of all allowable user spacecraft communications configurations based on links marked as active.
Note: This rule is unconstrained as to the number of elements in the generated set of configurations, and may easily give rise to a combinatorial explosion as the number of TDRSSs, the number of defined links, and the number of user antennas increase. Therefore, as these variables increase, the execution time of ECCO would be expected to increase dramatically.
- (4) TDRSS restrictions.
Remove any configuration which violates TDRSS restrictions:
-- a TDRS may not simultaneously support two links at the same frequency to the same user spacecraft.
-- a TDRS may not simultaneously support two MA forward links to the same user spacecraft.
-- a TDRS may not simultaneously support two SSA or two KSA links with the same TDRS antenna.
- (5) Optimization on priority links.
Remove any configuration such that the active priority link (if any) has a total BER margin less than the reference value. The reference value is determined as follows: examine the entries in the link combination table corresponding to the priority link, find the one which has the largest total BER margin, and reduce this value by 1 dB tolerance.
Note: In this and following rules, tolerance values are used to "soften" the discrimination between the configurations under consideration in order to reflect the fact that small differences in the computed BER

margin may not be significant and to permit subsequent rules to have some effect in the elimination process.

(6) Optimization on forward links.

Remove any configuration such that the sum of the BER margins of all forward links in that configuration is less than the reference value. This step is not performed if there are no active forward links. The reference value is determined as follows: for each configuration, sum the BER margins for all forward links in that configuration; the reference value is the largest such summed value reduced by 1 dB tolerance for each active forward link.

(7) Optimization on return links.

Remove any configuration such that the sum of the total BER margins (I-channel BER margin plus Q-channel BER margin) of all return links in that configuration is less than the reference value. This step is not performed if there are no active return links. The reference value is determined as follows: for each configuration, sum the total BER margins (I-channel BER margin plus Q-channel BER margin) for all return links in that configuration; the reference value is the largest such summed value reduced by 1 dB tolerance for each active return link.

(8) Optimization on links having the highest data rate on the priority channel.

(a) For each link defined in the link definition table, determine which channel is the priority channel based on the link type and the data group and mode.

(b) For each defined link, determine the data rate for the link's priority channel, and find the largest of these data rates.

(c) Determine which defined links have this data rate for their respective priority channels, and calculate a tolerance value equal to the number of such links times 1 dB.

(d) In each configuration, (i) find the sum of the BER margins for the priority channels having the highest data rate, (ii) find the largest such sum among all the configurations, and (iii) remove any configuration having such sum less than the largest such sum reduced by the tolerance value calculated in (c).

(9) Optimization on largest minimum BER margin.

For each configuration determine the smallest BER margin for all links in that configuration, and determine the maximum of such minimum margins. Remove any configuration having minimum margin less than that maximum.

(10) Optimum selection.

If more than one configuration remains, select one at random and designate it as the optimum configuration. Display the optimum configuration.

Note: This rule reflects the fact that all configurations uneliminated at this step are considered to be equally good, i.e., there exists no way to determine that one is any better than another.

F. ECCO Implementation

While this optimization problem may be attacked using a procedural language such as FORTRAN or C, certain considerations, including our desire to extend rule-based programming methodology into CLASS applications (with previous success at doing so [4]), led to employing the widely used rule-based programming language OPS5 [5] to build the experimental prototype discussed in this paper. OPS5 was considered superior in expressive power and in its ability to handle the potential combinatorial explosion entailed in generating communications configurations (see rule (2)). It should be noted that despite the use of a rule-based programming approach, many would not classify this prototype as a full expert system. This would be due mainly to the fact that it has no interactive interface for inputs from humans during execution, and does not operate on uncertain domain data.

To express the 10 optimization rules given above required approximately 110 OPS5 rules. Implementation of the prototype was carried out on a Hewlett-Packard Vectra personal computer (compatible with IBM PC/AT).

Fig. 3 presents the straightforward logical structure of ECCO. The block comprising the "elimination sequence" represents the majority of ECCO code as well as the majority of ECCO processing time.

As an example of the code in ECCO, Fig. 4 contains the OPS5 rules corresponding to rule (4) above concerning TDRSS restrictions.

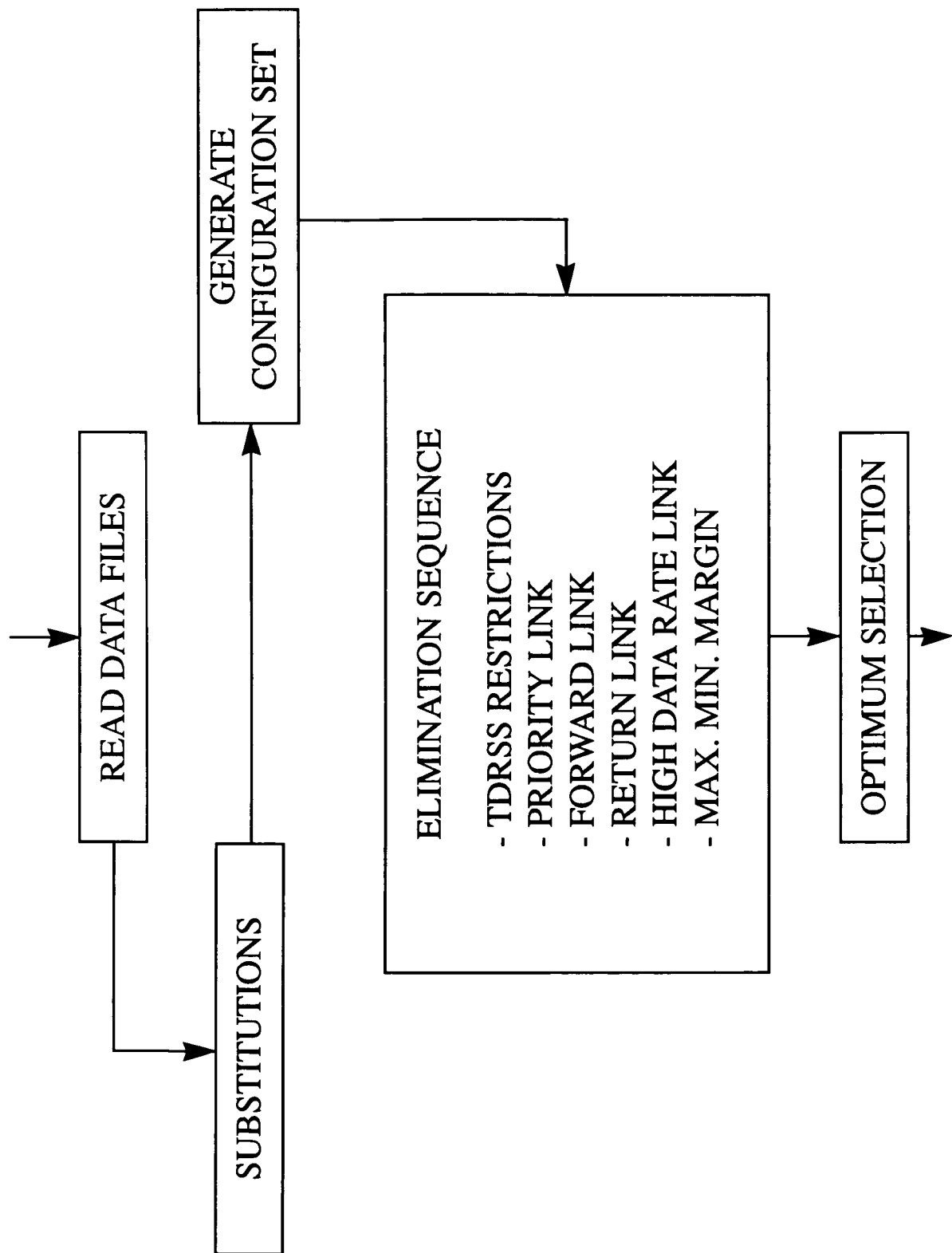


Fig. 3. Logical Structure of ECCO

TDRS COMMUNICATION LINK RESTRICTIONS

ORIGINAL PAGE IS
OF POOR QUALITY

This file contains the TDRS communication link restrictions.
1) can not have two links with same frequency to same TDRS
2) can not have two MA forward links to same TDRS
3) can not have two SSA or two KSA links to same SA antenna

```
(p RESTRICT:10:10-2-links-with-same-frequency-to-same-TDRS
; IF two links with same frequency to the same TDRS
; THEN remove configuration
(goal ^name restrictions)
(defined-link ^link-id <l>
  ^frequency <f> )
(defined-link ^link-id { <l1> <f> <l> }
  ^frequency <f> )
(link-combination ^link-id <l>
  ^sequence-num <s>
  ^supporting-tdrs <t> )
(link-combination ^link-id <l1>
  ^sequence-num <s1>
  ^supporting-tdrs <t> )
(uconfig-element ^uconfig-num <n>
  ^link-combination-sequence-num <s> )
( uconfig-element ^uconfig-num <n>
  ^link-combination-sequence-num <s1> )
-->
(make remove-uconfig <n> ))

(p RESTRICT:10:20-2-ma-forward-links-to-same-TDRS
; IF two ma forward links to the same TDRS
; THEN remove configuration
(goal ^name restrictions)
(defined-link ^link-id <l>
  ^service ma
  ^link-type f )
(defined-link ^link-id { <l1> <f> <l> }
  ^service ma
  ^link-type f )
(link-combination ^link-id <l>
  ^sequence-num <s>
  ^supporting-tdrs <t> )
(link-combination ^link-id <l1>
  ^sequence-num <s1>
  ^supporting-tdrs <t> )
(uconfig-element ^uconfig-num <n>
  ^link-combination-sequence-num <s> )
(uconfig-element ^uconfig-num <n>
  ^link-combination-sequence-num <s1> )
-->
(make remove-uconfig <n> ))
```

```
(p RESTRICT:10:30-2-ssa-or-2-ksa-links-to-same-sa-antenna
; IF two ssa or two ksa links of same type
; to the same sa antenna
; THEN remove configuration
(goal ^name restrictions)
(defined-link ^link-id <l>
  ^service <serv>
  ^link-type <type> )
(defined-link ^link-id { <l1> <f> <l> }
  ^service <serv>
  ^link-type <type> )
(link-combination ^link-id <l>
  ^sequence-num <s>
  ^supporting-tdrs <t>
  ^tdrs-antenna-id <ta> )
(link-combination ^link-id <l1>
  ^sequence-num <s1>
  ^supporting-tdrs <t>
  ^tdrs-antenna-id <ta> )
(uconfig-element ^uconfig-num <n>
  ^link-combination-sequence-num <s> )
(uconfig-element ^uconfig-num <n>
  ^link-combination-sequence-num <s1> )
-->
(make remove-uconfig <n> ))

(p RESTRICT:10:50-no-valid-uconfigs
; IF there are no valid uconfigs
; THEN stop
(goal ^name restrictions)
-(uconfig-element)
-->
(write out (crlf) (crlf))
(write out | No possible user configuration due to |
  | TDRS restrictions. |)
(write out (crlf) (crlf))
(closefile out)
(halt))

(p RESTRICT:10:60-change-goal-warnings
; goto warnings (WARNINGS.OPT)
{ (goal ^name restrictions) <goal> }
-->
(modify <goal> ^name warnings ))
```

FIG. 4. OPS5 Rule Corresponding to Rule 4 (TDRSS Restrictions).

TABLE I
Defined Link Table

Link ID	Status	Service Category	Type	Data Rate (Bits Per Second)		Frequency (Hz x 100)	Data Group	Mode	Priority Level
				I Channel	Q Channel				
H1	Inactive	SSA	R	32000	32000	22875	1	1	Nonpriority
H2	Active	SSA	R	32000	512000	22875	1	3	Nonpriority
L3	Active	SSA	R	1000	1000	22875	1	1	Nonpriority
H4	Inactive	MA	R	1000	32000	22875	1	1	Nonpriority
L5	Active	SSA	F	1000	0	21064	0	0	Priority
L6	Active	SSA	F	125	0	21064	0	0	Priority
L7	Inactive	MA	F	125	0	21064	0	0	Priority
L8	Inactive	MA	F	1000	0	21064	0	0	Priority

Note: SSA - S-Band Single Access MA - Multiple Access R - Return Link F - Forward Link

TABLE II
Link Substitution Table

Link ID	Substitute Link ID
H1	H4
H1	H2
L5	L7
L6	L7

TABLE III
Link Combination Table

Sequence Number	Link ID	User Antenna ID	Supporting TDRS ID	TDRS Antenna ID	BER Margin (dB)		
					I Channel	Q Channel	I + Q Total
1	H1	HIGH	E	W	0	10	10
2	H1	HIGH	W	W	9	11	20
3	H1	HIGH	W	E	8	10	18
4	H2	HIGH	W	E	21	11	32
5	H2	HIGH	W	W	12	22	34
6	L3	OMNI	W	E	23	13	36
7	L3	OMNI	W	W	21	15	36
8	L3	OMNI	E	E	5	8	13
9	L3	OMNI	E	W	7	10	17
10	L5	HIGH	W	W	12	0	12
11	L5	HIGH	W	E	7	0	7
12	L7	HIGH	E	M	10	0	10
13	L7	HIGH	W	M	5	0	5

Note: OMNI - Low Gain Antenna M - Multiple Access E - East W - West
HIGH - High Gain Antenna

IV. TEST CASE

The experimental version of ECCO has been tested using input data based on telecommunications system specifications for a planned scientific mission [6].

Input data for an illustrative test case is shown in Tables I, II, and III, showing, respectively, the defined link table, the link substitution table, and the link combination table for a hypothetical mission time point.

When ECCO is executed using this input data, the rule on substitutions (optimization rule (2)) will be invoked because there is no entry in the link combination table for defined link L6, indicating that FPS found no acceptable (i.e., nonnegative) BER margin for link L6 at the mission time point in question; however, as provided in the defined link table, this link is scheduled to be active at that time. The link substitution table shows that link L7 can be substituted for L6. Since the link combination table shows acceptable BER margin for this substitute and since L7 is not active, the substitution is made. When the substitution is made the status of L6 is changed from active to inactive, and the status of L7 is changed from inactive to active.

Applying the definition of user communications configuration given earlier (Section II. B.), we see that 32 different configurations are possible using the four defined links (links H2, L3, L5, and L7) that are now active following action by the link substitutions rule (optimization rule (2)). Each of the following groupings of four link combination sequence numbers represents one of these 32 possible configurations as generated by optimization rule (3) (also see Table IV):

5	5	5	5	5	5	5	5	
9	9	9	9	8	8	8	8	...
11	11	10	10	11	11	10	10	
13	12	13	12	13	12	13	12	

As shown in Table IV, twenty-four of these alternative configurations are eliminated by optimization rule (4) due to violations of TDRSS restrictions. Optimization rule (5) concerning priority links eliminates four more, leaving only four configurations. None are eliminated by optimization rule (6) (forward links), but each of rules (7), (8), and (9) eliminates one. Since only one alternative configuration then remains (comprising sequence numbers 5, 9, 10, and 12), it is designated as the optimum (see Test Case Output in Fig. 5).

SUBSTITUTED LINKS

Due to unacceptable EIRP margin at time 1
OPTIMIZER required the following link substitutions

link-id 17 for link-id 16

OPTIMUM CONFIGURATION

The OPTIMUM CONFIGURATION is user configuration 4
It has the following links ...

SEQ-NO	LINK-ID	USER-ANT	TDRS	TDRS-ANT	I-FWD-BER	Q-BER
9	13	omni	e	w	7	10
10	15	high	w	w	12	0
12	17	high	e	m	10	0
5	h2	high	w	w	12	22

Fig. 5. Test Case Output.

TABLE IV
LINK COMBINATION ELIMINATIONS FOR TEST CASE.
A COMBINATION ELIMINATED BY A RULE IS INDICATED
BY "X" IN THE COLUMN FOR THAT RULE.
THE DISCOVERED OPTIMUM IS INDICATED BY "O"

Configur- ation #	Link Combination Sequence Numbers In Configuration	OPTIMIZATION RULE						
		TDRSS Restrictions	Priority Link	Forward Link	Return Link	High Data Rate Link	Max Min Margin	Optimum Selection
1	13 11 9 5	X						
2	12 11 9 5		X					
3	13 10 9 5	X						
4	12 10 9 5							O
5	13 10 8 5	X						
6	12 11 8 5		X					
7	13 10 8 5	X						
8	12 10 8 5						X	
9	13 11 7 5	X						
10	12 11 7 5	X						
11	13 10 7 5	X						
12	12 10 7 5	X						
13	13 11 6 5	X						
14	12 11 6 5	X						
15	13 10 6 5	X						
16	12 10 6 5	X						
17	13 11 9 4	X						
18	12 11 9 4		X					
19	13 10 9 4	X						
20	12 10 9 4					X		
21	13 11 8 4	X						
22	12 11 8 4		X					
23	13 10 8 4	X						
24	12 10 8 4				X			
25	13 11 7 4	X						
26	12 11 7 4	X						
27	13 10 7 4	X						
28	12 10 7 4	X						
29	13 11 6 4	X						
30	12 11 6 4	X						
31	13 10 6 4	X						
32	12 10 6 4	X						

Hand checking shows that the test case output is consistent with the optimization rules delineated in Section III. For example, consider rule (4) concerning TDRSS restrictions (also see the actual OPS5 code in Fig. 4). One of the restrictions stipulates that there may not be two links at the same frequency between any user spacecraft and any TDRS. By inspection of the data in Tables I through IV, we see that this restriction eliminates any configuration containing any one of the link combination pairs (4,6), (4,7), (5,6), (5,7), (10,13), and (11,13). Also, it is noted that any configuration containing either of the link combination pairs (4,6) and (5,7) is also eliminated by the third TDRSS restriction listed under optimization rule (4) (a TDRS may not simultaneously support two SSA or two KSA links with the same TDRS antenna).

A number of additional test cases based on other missions have been devised and found consistent with the rules. Exhaustive testing remains to be done, but results to date support the conclusion that the rule-based approach used in ECCO is valid.

V. CONCLUSION

The experimental rule-based system ECCO has been used to demonstrate the validity of using a rule-based systems approach to the problem of finding the optimum user spacecraft communications configurations at any isolated time (single point optimization) based on successive eliminations of non-optimum configurations (incremental optimization method). Successful demonstration of the ECCO prototype is an important step toward incorporation of rule-based programming methodology into CLASS applications.

Future development of ECCO will focus on the following areas: optimizing communications performance by means of user spacecraft attitude adjustments, optimizing while precluding user antenna toggling, and multipoint optimization relative to mission phase. Optimum communications performance will also be extended to include stochastic losses (RFI, multipath, etc.) and synchronization (e.g., probability of loss of lock).

ACKNOWLEDGMENTS

The authors wish to thank CLASS Project Manager R. D. Godfrey of the NASA/Goddard Space Flight Center for originating the concept of a spacecraft communications configuration optimizer and for lending his knowledge concerning TDRSS. We are also grateful to our supervisor, F. J. Stocklin, for his encouragement in our writing of this paper.

REFERENCES

- [1] *Tracking and Data Relay Satellite System Users' Guide*, Revision 5, NASA Goddard Space Flight Center, STDN No. 101.2, September, 1984.
- [2] W. R. Braun and T. M. McKenzie, "CLASS: A Comprehensive Satellite Link Simulation Package", *IEEE Journal on Selected Areas in Communications*, Vol. SAC-2, No. 1, January, 1984.
- [3] R. D. Godfrey, "CLASS Requirements and Development Document", CLASS Document No. 310.0, NASA Goddard Space Flight Center, October 1985.
- [4] J. L. Rash, "A Prototype Expert System in OPS5 for Data Error Detection", *Telematics and Informatics*, Vol. 3, No. 3, 1986.
- [5] L. Brownston, R. Farrell, E. Kant, and N. Martin, *Programming Expert Systems in OPS5, An Introduction to Rule-Based Programming*, Addison-Wesley Publishing Company, Inc., 1985.
- [6] "Radio Frequency Interface Control Document Between Gamma Ray Observatory and the Tracking and Data Relay Satellite System", NASA Goddard Space Flight Center, November, 1986.

Integrated Resource Scheduling in a
Distributed Scheduling Environment

by

David Zoch and Gardiner Hall
Ford Aerospace Corporation
College Park, MD

ABSTRACT

The Space Station era presents a highly-complex multi-mission planning and scheduling environment exercised over a highly distributed system. In order to automate the scheduling process, customers require a mechanism for communicating their scheduling requirements to NASA. An expressive scheduling notation that captures a wide range of customer requirements and scheduling options is one solution to this problem.

The NASA planning and scheduling environment is itself distributed: The SSIS ADD defines at least six types of elements that will play a major role in the planning and scheduling process (e.g., PSC, SSSC, NSTS MCC, POCC's etc.). Each of these elements is responsible for the creation and maintenance of schedules that are related (via shared resources, for instance). These schedules need to be integrated in such a way that inconsistencies are resolved. An important step in this schedule inconsistency resolution process is the identification and definition of the inter-scheduler messages that will be needed by each scheduler.

This paper describes a request language that a remotely-located customer can use to specify his scheduling requirements to a NASA scheduler, thus automating the customer-scheduler interface. This notation, which we have nicknamed FERN (Flexible Envelope-Request Notation), allows the user to completely specify his scheduling requirements such as resource usage, temporal constraints, and scheduling preferences and options. FERN also contains mechanisms for representing schedule and resource availability information, which are used in the inter-scheduler inconsistency resolution process.

Additionally, this paper describes a scheduler that can accept these requests, process them, generate schedules, and return schedule and resource availability information to the requester. The Request-Oriented Scheduling Engine (ROSE) has been designed to function either as an independent scheduler or as a scheduling element in a network of schedulers. When used in a network of schedulers, each ROSE communicates schedule and resource usage information to other schedulers via the FERN notation, enabling inconsistencies to be resolved between schedulers. Individual ROSE schedules are created by viewing the problem as a constraint satisfaction problem with a heuristically guided search strategy.

INTRODUCTION

The Space Station/CDOS era presents a highly-complex planning and scheduling environment with many new challenges. Telescience, an operations concept that allows users to control their instruments from their home institutions must be supported. As part of this approach, telescience users need a distributed and hierarchical planning and scheduling capability that parallels the architecture of the Space Station and associated elements.

The scheduling process, which is currently manual with some computer assistance, must become highly automated. This requires that user scheduling requirements be encoded in such a way that they can be understood by an automated NASA scheduling system. Two approaches are possible. One approach is to encode the knowledge used to plan the activities of each instrument in a knowledge-based system and build this knowledge into the scheduler. This approach has been successfully demonstrated in the Mission Operation Planning Assistant (MOPA) as being feasible for instruments on board the UARS (Upper Atmospheric Research Satellite). In this approach, the scheduler knows what each instrument is trying to accomplish, and the appropriate instrument operating modes and requirements necessary to support a specific data-gathering activity. It keeps track of the observations that have been made by each instrument and can re-plan the instrument's operations in case of an unexpected target of opportunity.

This approach has three major drawbacks: A sizable knowledge-engineering task is required in order to specify the planning and scheduling requirements of each instrument. Secondly, this knowledge-engineering must be repeated for each new instrument or to support changing scientific objectives. Thirdly, and most importantly, the approach places a great deal of responsibility on the NASA scheduling system, making it responsible for the operation of the instruments, which is contrary to the concept of telescience, in which users are responsible for instrument operations.

A more promising approach to support multiple distributed users is the telescience approach in which users operate their instruments from their home institutions and submit requests to a NASA scheduler specifying their scheduling support requirements. Scientists then receive an allocation of resources in what is commonly called the "resource envelope". This approach has several advantages (1) the NASA scheduling system is simplified, since it is only responsible for allocating "resource envelopes" to each user (2) since resource envelopes are being scheduled instead of individual commands, there are fewer items to schedule, (3) the re-scheduling problem is simplified, (4) users are free to operate their instruments as they choose, as long as the experiment can be

performed within the allocated "resource envelope", and (5) No additional knowledge engineering or software changes have to be made when a new payload is added or suddenly takes on a new objective.

An additional complication in the Space Station planning and scheduling architecture is that there are many schedules involved in the overall process; these schedules will be developed by different organizations within NASA (at possibly different locations) with different scheduling objectives. Generating one large schedule is not feasible. Unfortunately, the schedules of one organization will impact other schedules and, therefore, the different schedulers must communicate scheduling information to each other in order to resolve "inter-scheduler conflicts" (For instance, if scientists want to make a coordinated observation using two instruments that have schedules that are generated by different schedulers then the two schedulers need to coordinate in order to ensure that the observations are scheduled at the same time.

PROBLEM

As previously mentioned, Space Station presents many new planning and scheduling challenges. The goal of our prototyping effort is to demonstrate a planning and scheduling environment that shows the essential concepts needed to support the Space Station Customer Data and Operations System (CDOS).

After choosing a distributed architecture (see Figure 1) as recommended in [FAC,1987,1], it was apparent that the problem could be divided into three steps:

1. Define the types of scheduling information that need to be communicated between schedulers and between scientific users and NASA schedulers.
2. Implement a scheduler that can accept and process these "scheduling messages" and send appropriate messages to other schedulers.
3. Build a network of these schedulers, in order to demonstrate a distributed scheduling environment.

Figure 1 shows a simplified distributed scheduling architecture. Users send requests (scheduling messages) from any remote location (Instrument Planning Software) to any one of a number of PRMC's (Payload Resource Management Centers) where schedules are created for a specific group of instruments. Schedule information is then returned to the user.

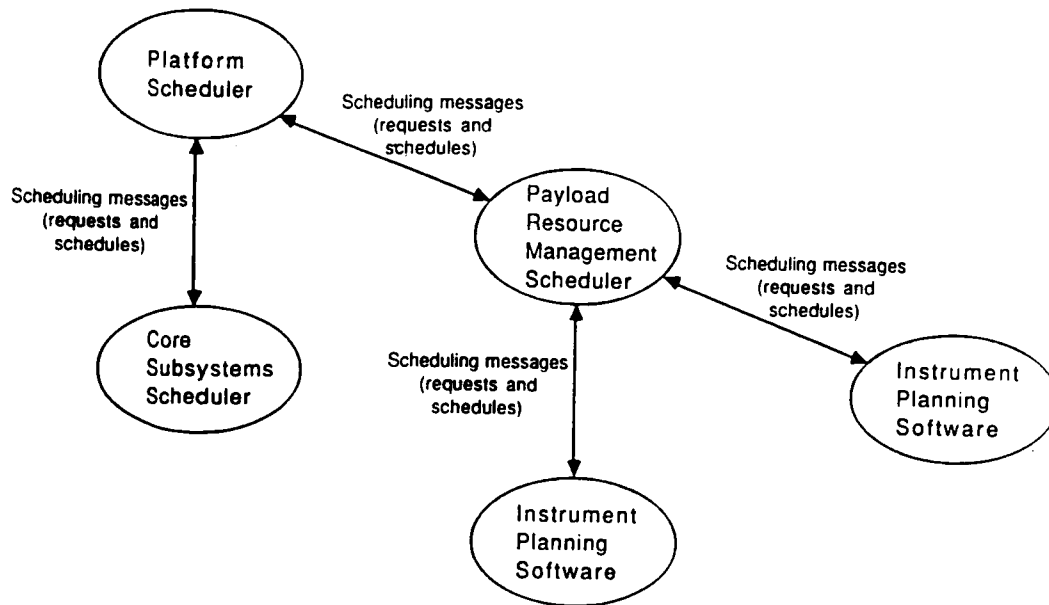


Figure 1. A Distributed Scheduling Architecture

Similarly, each PRMC sends scheduling messages to request resources from its parent node, the Platform Scheduler, and receives resources. Temporal Constraint information (information concerning experiment coordination and sequencing) is also communicated at each level in the hierarchy.

In order to automate the first step in the scheduling process, accepting user requests for the scheduling of an instrument, a "request language" must be defined in which a scientific user can express all of his scheduling requirements.

FERN (Flexible Envelope-Request Notation) is the request language that resulted from a joint effort between Ford Aerospace and the scientists who control the SME (Solar Mesospheric Explorer) at the University of Colorado LASP (Laboratory for Atmospheric and Space Physics). FERN allows the user to completely specify his scheduling requirements such as resource usage, temporal constraints (request sequencing and coordination), request priority, preferences, alternatives, and environmental impacts. Additionally, FERN provides several user-oriented convenience features, such as allowing the user to specify repetitive requests, for instance, "Schedule this request twice per day". While primarily designed for communicating information between the scientific user and the PRMC (i.e., a NASA automated scheduler) the resulting scheduling messages are also useful for expressing scheduling requirements and schedules throughout the entire distributed

scheduling system.

The Request-Oriented Scheduling Engine (ROSE) is a scheduler that demonstrates the feasibility of accepting and processing requests in the FERN notation. ROSE was designed to function as a general scheduler, and thus can be used at each of the NASA scheduling nodes in Figure 1 (i.e., PRMC scheduler, Platform Scheduler, Core Subsystems Scheduler). This paper describes the features of the FERN request language and the ROSE scheduler.

SCHEDULING LANGUAGE FEATURES

The FERN request language was designed to provide a scientist user with a mechanism to specify all of his scheduling requirements. The FERN "Preliminary Request" message is used to make this initial specification (there is also a "Refined Request" message that is used later in the scheduling process). Additionally, FERN was designed to allow the user to specify these requirements in a manner that is consistent with the way users think about payload scheduling. For instance, users often perform an experiment that consists of several different steps, or "phases" each of which might have varying resource requirements. FERN supports this by allowing multiple phases with varying resource levels to be specified within the request.

The following types of information can be represented in the REQUEST message:

1. Resource Requirements -- specifies any number of "phases" for a request and the duration and resources needed for each phase (described in more detail below).
2. Temporal Constraints -- specifies where this request can be scheduled in relation to orbital or other events (for example, "schedule this request within 3 minutes of an equator crossing"). Also used to specify experiment sequencing, e.g., schedule REQUEST A before scheduling REQUEST B
3. Priority -- specifies a measure of the importance of this request to the user on a scale of 1 (not critical) to 10 (very important).
4. Identification Information -- specifies the sender of the message, a name, and the time that the message was sent.
5. Repetitive Scheduling Information -- specifies that

a request is to be scheduled multiple times, for instance, "Schedule this request 2 times per day."

6. Preference Information -- specifies a user temporal reference. A temporal preference is similar to a temporal constraint except that it places no actual restriction on the scheduling of the envelope. For instance "Schedule this request as close to Wednesday as possible" specifies a preference, while "Schedule this request before Wednesday" specifies a temporal constraint.

7. Environmental Requirements/Impacts -- specifies (1) the possibly negative impacts that this experiment will have on other experiments, for instance, causing excessive amounts of vibration, and (2) the environmental requirements needed in order to schedule this request

Many of these user scheduling requirements are viewed as being flexible and can be relaxed in one way or another. For instance, FERN allows the user to specify a desired quantity of a resource and also a minimally acceptable amount.

Figure 2 shows a screen image of a pretty-printed request message (the underlying request language uses a LISP-like syntax). The request shown has three phases (which will be performed contiguously) each of which specifies the resource requirements for that phase of the experiment. Two types of requirement relaxation are shown in this request: phase duration relaxation and resource relaxation. In phase 1 of the resource section, the user has requested that the desirable duration of this phase is 48 minutes, but that if it makes a difference in getting the request scheduled, 36 minutes is sufficient. Also in phase 1, the user has specified that 10 units of power is the nominal amount required, but 4 is sufficient. A user is not required to specify any relaxation amounts but can do so in order to increase the probability that his request will get scheduled.

Temporal Constraint Notation





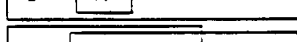
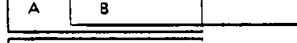

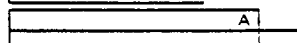
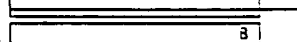
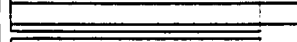
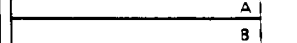
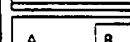
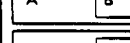
Temporal Constraints are used in request messages to express the desired relationship of the scheduled request to pre-defined orbital (or user-defined) events (for instance, "spacecraft day") or to other requests. "Schedule this request any time on Wednesday" and "Schedule this request any time after request SOLAR-OBSERVE-3 is scheduled" are both examples of temporal constraints that can be specified in a request.

REQUEST-ORIENTED SCHEDULING ENGINE						
Process Operations		Communications Interface				
Initialize System	PRMC Init	PROCESS	RUN-STATE	Z-STATE	STP-RESN	ACTIVITY
Send Test Messages	stop prmc	*DDMC*	DIININTMC	Sleep	---	---
Schedule Request						
From: ICDC.1						
To: PRMC						
Message Type: PRELIM-REQUEST						
Time Sent: 3/05/95 12:30:00						
Name: 2-ORBIT-SCAN 1						
Request Priority: 1.0						
Preference: Schedule as soon as possible						
Repeat: Schedule this request 4 times every 2 days 16 minutes						
Resource Envelope Phases:						
Phase 1						
Duration: 48 minutes (MINIMUM of 36 minutes)						
POWER Nominal: 10 Minimum: 4						
UV-SPEC 1						
COMMAND-LINK 8						
HIGH-RATE-RETURN-LINK 100						
Phase 2						
Duration: 48 minutes						
POWER 5						
UV-SPEC 1						
COMMAND-LINK 8						
Phase 3						
Duration: 1 hour 36 minutes						
POWER 10						
UV-SPEC 1						
COMMAND-LINK 8						
HIGH-RATE-RETURN-LINK Nominal: 100 Minimum: 80						
Temporal Constraints:						
Schedule	AFTER	*MONDAY*				
Schedule	BEFORE	*THURSDAY*				
Schedule	DURING	*SPACECRAFT-NIGHT*				
Schedule	AFTER	COORDINATED-PROBE-1				

Figure 2. A formatted Request Message

Figure 3 gives a pictorial representation of the possible relationships between two time intervals and the notation FERN uses to express the relationship. (The request formatter translates these symbols in the request language to words such as "before," "after," and "during" as in Figure 2.) Each temporal relation consists of three symbols: the first defines the relationship between the start times of the two intervals, the third defines the relationship between the end times of the two intervals, and the second describes the overlap between the two intervals ("o" means there is some overlap, "x" means there is no overlap, and "!" means that the intervals abut each other.) Using the above symbols in conjunction with the "don't care" symbol, "-", and the standard numeric comparison operators, ("<", ">", "=" etc.) provides a powerful notation for concisely expressing the temporal interval relations shown in Figure 3.

The temporal constraint portion of the request message is also used to specify scheduling alternatives using some additional operators. For instance, the notation "A xxx B" specifies that requests A and B are "mutually exclusive," i.e., that either one or the other (or neither) of them should be scheduled, but not both. By assigning a higher priority to request A, a user has effectively specified that request B should be scheduled if request A cannot be scheduled (since the scheduler will naturally attempt to satisfy the higher priority request first).

TEMPORAL RELATIONSHIP		REPRESENTATION
1		A <X< B
2		A >X> B
3		A <0> B
4		A >0< B
5		A <0< B
6		A >0> B
7		A =0< B
8		A =0> B
9		A =0= B
10		A <0= B
11		A >0= B
12		A <!=< B
13		A >!=> B
1'	A STARTS BEFORE B STARTS	A <-- B
2'	A STARTS AT THE SAME TIME AS B	A == B
3'	A STARTS AFTER B	A >-- B
4'	A ENDS BEFORE B ENDS	A --< B
5'	A ENDS AFTER B ENDS	A --> B
6'	A ENDS AT THE SAME TIME AS B	A == B
7'	A OVERLAPS B	A -0- B
8'	A DOES NOT OVERLAP B	A -X- B

KEY			
SYMBOL	INTERPRETATION	SYMBOL	INTERPRETATION
X	DOES NOT OVERLAP	<	BEFORE
0	OVERLAPS	>	AFTER
!	ABUTS	-	DON'T CARE
=	SAME TIME		

Figure 3. Temporal Relation Operators

ROSE CAPABILITIES

ROSE is currently under development as a tool to demonstrate a scheduling system that supports telescience concepts. The current major capabilities of ROSE are (1) to receive scheduling messages via a file transfer protocol from any machine located on the host network and respond with appropriate scheduling messages, (2) to create an initial schedule from these requests, and (3) to reschedule (if appropriate) in order to satisfy local scheduling goals.

ROSE was originally implemented on a Texas Instruments Explorer and has been ported to the Symbolics 36xx environment under releases 6.1 and Genera 7.

Communications Capabilities

ROSE supports inter-scheduler communication through the transmission of resource requests and scheduled resource data. Users transmit requests to ROSE, described in the FERN notation. The user receives two responses from ROSE. The first is an acknowledgment of the message, confirming receipt of the message by ROSE. After ROSE completes processing of the request it will transmit one of three responses to the user. The first possible response ROSE generates is a scheduled request message. Contained in this message is the name of the scheduled request, the time assigned to the request, and the resource levels dedicated to the request. The second possibility is a scheduling failure message. The third possibility is an erroneous request message, indicating an improperly formed request.

Scheduler Capabilities

"Scheduling" in the ROSE system is the ability to create an initial schedule from a set of requests and scheduling heuristics. Selection of scheduling heuristics allows creation of alternative resource schedules from the same requests. This capability provides several advantages in both the operational and development modes. The primary advantage is the ability to tailor the scheduling system to the current mission environment. The Space Station environment is dynamic in nature, due to changing mission objectives, equipment deterioration, and targets of opportunity. This flexibility provides a mechanism for responding to short term changes of standard operating procedures by allowing preplanned rules of scheduling to be defined. For example, if a shortage of a resource is anticipated for a short period, ROSE can create schedules optimizing that resource. ROSE also aids in the development of scheduling heuristics by allowing comparison of different schedules. The final advantage is the ability to use ROSE as a drawing board for performing "what if" scheduling by

mixing request selection and placement strategies.

Rescheduling Capabilities

It is unlikely that an initial schedule can be created that satisfies all of the requests all of the time. Conflicts between requests will frequently occur. Rescheduling will usually be a necessary step after the initial schedule is created. In a simple, centralized scheduler the only way to resolve conflicts is to choose the higher priority request. In a network of ROSE schedulers, each allowing flexible requests, there are several options:

1. Overbook the resource -- in our distributed scheduling environment, overbooking is a viable conflict resolution scheme since additional resources can potentially be acquired from another scheduler.
2. Relax this request -- a minor adjustment to the scheduling requirements of the request might allow it to be scheduled.
3. Relax other requests -- if it is important to schedule more requests, higher priority requests might have their requirements relaxed in order to accommodate lower priority requests.
4. Acquire additional resources -- In a network of communicating schedulers, it might be desirable to actually request and obtain resources from another scheduler
5. Simply choose the higher priority request.

Four of these five re-scheduling options are currently available in ROSE. In the next release, we will implement a network of ROSE schedulers, allowing implementation of strategy 4, and the automatic selection of the appropriate re-scheduling strategy. Currently, rescheduling options must be manually selected.

User Interface

The user interface consists of four screens containing ROSE command menus and scheduling data. Each screen logically groups the various scheduling operations and data simplifying user interaction with the system. The integration of graphics and text presents request and schedule data in an easily understood manner. Interaction with the system is performed almost exclusively through mouse/menu operations reducing operator typing. ROSE provides a Communications screen, a Scheduler screen, a Resource/Schedule screen, and an Unscheduled Request screen. A description of each screen follows.

Communications Screen

The Communications screen (Figure 4) provides a top level view of the user network. The screen is divided into four areas: the Process Operations menu, the Process Monitor window, the Scheduling Messages window, and a Notifications window. The Process Operations menu, located in the upper left hand corner of the screen, presents the available "housekeeping" operations. The center of the Communications screen contains the Scheduling Messages window, displaying all scheduling messages received and transmitted by ROSE. The remainder of the screen contains the Notifications window, allocated for ROSE system generated messages to the ROSE operator.

The Scheduling Messages window contains all active messages received and transmitted by ROSE. The message display provides type, origination, destination, and name data for the message. If the user desires, mouse selection of the message will display the entire message in a pop-up window. Scrolling through the Scheduling Messages window allows the Rose user to view all messages.

Scheduler Screen

The Scheduler screen (Figure 5) provides schedule generation commands and displays. This screen is useful from a development point of view because it provides all operations necessary for the creation of a schedule. The left hand portion of the screen contains the five Scheduler Options menus, useful for tailoring a scheduling strategy. Located to the right of the Scheduler Options menus is the Alternative Schedules window. This feature aids system developers by displaying schedule statistics for a scheduling run. The Alternate Schedule portion of the screen is divided into three areas allowing the comparison of three schedules simultaneously.

ORIGINAL PAGE IS
OF POOR QUALITY

REQUEST-ORIENTED SCHEDULING ENGINE							
Communications Interface							
Process Operations		PRMC Inq	PROCESS	RUN-STATE	Z-STATE	STP-RESN	ACTIVITY
Initiate System		stop prmc	*PRMC*	RUNNING	Sleep	---	---
Send Test Messages		re-start prmc	*MESSAG	RUNNING	End-loop	---	---
stop cam		Run History	*ICDC_1	RUNNING	Sleep	---	---
re-start cam		Set Request Input file	*ICDC_2	RUNNING	Sleep	---	---
Enable/Disable Sound		Load Resources					
Set Resource File		Delete Name					
Enable Request Loading							
Save							
Process Monitor Window							
Incoming Scheduling Messages							
#	TYPE	FROM	TO				
0	INITIALIZE-SCHEDULING-INTERVALS	:LOCAL	:PRMC	LASP			
0	SET-SCHEDULING-PERIOD	:LOCAL	:PRMC	1			
1	DEFINE-INTERVAL-SET	:ICDC_1	:PRMC	*EARLY-IN-WEEK*			
1	DEFINE-INTERVAL-SET	:ICDC_1	:PRMC	*WEDNESDAY-NIGHT*			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	SUN-OBSERVE			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	DEEP-SPACE-PROBE			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	SUN-LIMB-SCAN			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	EARTH-LIMB-SCAN			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	STAR-SCAN			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	*-PROBE			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	H-PROBE			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	SUN-SCAN			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	STAR-SCAN			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	CALIBRATE			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	FINE-CALIBRATE			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	G-PROBE			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	T-PROBE			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	U-PROBE			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	R-PROBE-X			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	COORDINATED-PROBE			
1	PRELIM-REQUEST	:ICDC_1	:PRMC	2-ORBIT-SCAN			
RESOURCE-FILE set to 0<LNFS-PATHNAME "MERCURY:>irs>symbolics>data>day001-day007.reso.1"> *COM-IN-FILE* set to 0<LNFS-PATHNAME "MERCURY:>irs>symbolics>data>day001-day007.reqn.1"> *RESOURCE-FILE* set to 0<LNFS-PATHNAME "MERCURY:>irs>symbolics>data>day001-day007.reso.1"> Loading MERCURY:>irs>symbolics>data>day001-day007.reso.1 into package IRS							
NOTIFICATIONS							

Figure 4
Communications Screen

ORIGINAL PAGE IS
OF POOR QUALITY

REQUEST-ORIENTED SCHEDULING ENGINE		
Scheduler Interface		
Scheduler Option Menus	Alternative Schedules	
Select Scheduler Requests	Schedule A	
Week 1 Week 2 Week 3 Reset Requests	Scheduling Week: Week 1 Selection Strategy: Maximum Temporal Constraints Placement Strategy: First Opportunity Number of Scheduled Requests: 190 Number of Unscheduled Requests: 41	
Scheduler Commands	Select	
Expand Requests Create Preliminary Schedule Reschedule Unscheduled Requests Reschedule All Requests Unschedule All Requests	Schedule B	
Selection Strategies	Scheduling Week: Week 1 Selection Strategy: Maximum Peak Resource Utilization Placement Strategy: Best Temporal Fit Number of Scheduled Requests: 192 Number of Unscheduled Requests: 39	
Maximum Temporal Constraints Maximum Static Constraints Maximum Peak Resource Requirements Maximum Cumulative Resource Requirements Maximum Total Resource Requirements	Select	
Placement Strategies	Schedule C	
First Opportunity Best Temporal Fit Best Resource Fit Bypass User Preferences	Scheduling Week: Week 1 Selection Strategy: Maximum Peak Resource Utilization Placement Strategy: Best Temporal Fit Number of Scheduled Requests: 69 Number of Unscheduled Requests: 182	
Rescheduling Strategies	Select	
Change Overbooking Limits Relax Resource Requirements Relax Temporal Constraints		
69 Requests Scheduled, 162 Remaining Strategy used was :COMPACT-PREF 162 Requests went unscheduled Scheduled task R-PROBE-X, # 12 at time 330400		Select Display Communications Scheduler Schedule/Resource Unscheduled Requests
Messages		

Figure 5
Scheduler Screen

Resource/Schedule Screen

The Resource/Schedule screen (Figure 6) provides a graphical representation of schedule data. The top half of the screen presents a normalized plot of remaining resource data. The resource plot is useful because it allows the user to determine the amount of resource utilization. The bottom half of the screen displays the generated schedule in a timeline format. The timeline is an easily used display to verify a schedule.

A mouse click within the resource plot portion of the screen presents the user with a pop-up menu. Possible user selections are: view unscheduled requests, or choose new resource parameters to be plotted. By viewing an unscheduled request and plotting the most constraining resources the user can determine a close resource fit for the request. Depending on the closeness of the fit the user may take the appropriate scheduling action to accommodate the request.

Using the timeline presentation the ROSE user can verify a schedule. The scale of the timeline is user selectable, providing the capability to zoom in on a time period of interest. If the timeline is too large to fit into the time scale selected, the user can scroll through the timeline. Individual scheduled requests are selectable allowing user review of the fine detail of a request.

Unscheduled Requests Screen

The Unscheduled Requests screen (Figure 7) displays a list of requests that could not be scheduled due to resource conflicts or unsatisfiable temporal constraints. The top portion of the screen displays all unscheduled requests and an indication of the reason(s) for the failure to schedule that request. The most constraining resource is given for each request, along with an indication of how much this specific resource limits the scheduling of the request ("moderate" indicates that the request is constrained to 50% of the entire week-long schedule, "high" indicates 20%, etc.). An indication is also given of how much the request's temporal constraints restrict its placement on the schedule. For example, the highlighted request in Figure 7, "2-ORBIT-SCAN", is primarily limited by its need to use the HIGH-RATE-ANTENNAE. This display provides a good indication of overall resource shortages.

The bottom portion of this screen is used to plot available start times for a request with respect to user-selected resources. The user can select any combination of resources. The plot in Figure 7 shows that (1) the "command link" resource did not limit the scheduling of this request, (2) the temporal constraints placed on this request limited its scheduling to one three-day window near the middle of the schedule, and (3) sufficient power was available during six intervals. The RESULT (the intersection of the other three plots) shows where

[illegible]

Figure 6
Resource/Schedule Screen

ORIGINAL PAGE IS
OF POOR QUALITY

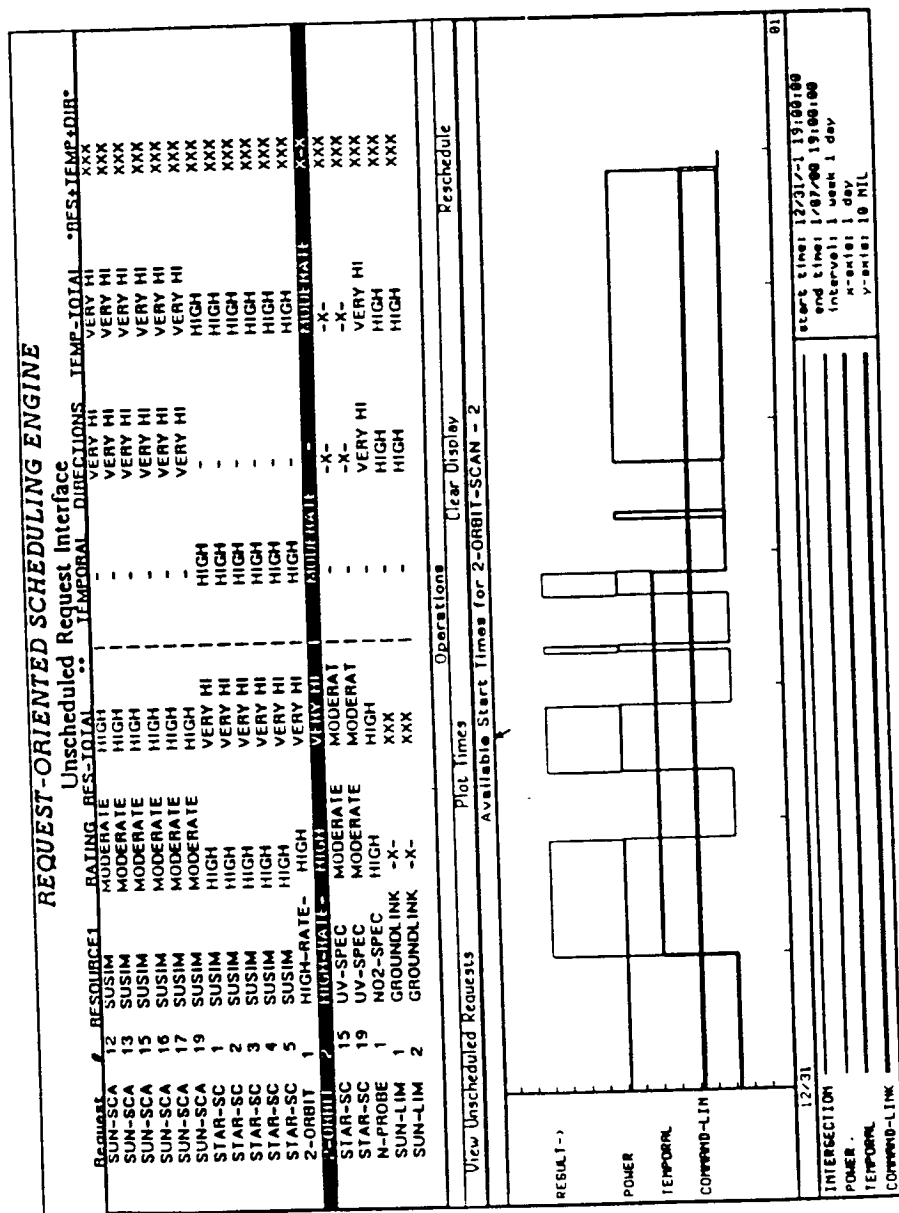


Figure 7
Unscheduled Request Screen

this request can be scheduled with respect to these two resources and its temporal constraints. It is also possible to plot a graph that shows where the request can be scheduled with respect to all of its resource requirements.

SUMMARY

We have demonstrated the effectiveness of FERN by encoding a wide range of requests for experiments on the SME spacecraft. The "smart request language" approach has many benefits that will be important in the Space Station planning and scheduling environment:

1. The user is allowed to specify a wide range of requirements and scheduling options that are not typically supported by schedulers that use simpler mechanisms such as tables or data records.
2. The user can tailor the language to his application in a manner similar to STOL and CSTOL (we don't expect the user to generate request manually, we expect that the user's COMPUTER will generate the requests; the user is free to define any interface to the scheduling language).
3. Provides high functionality -- for instance, users may state requirements in terms of orbital, atmospheric, solar, stellar, etc. events instead of window start and stop times. Users can also define their own "events".
4. Provides a consistent interface across Space Station elements
5. Provides users with "smart" requests which will reduce the number of messages sent and provide increased security. (Since users specify their requirements instead of a specific start time, users do not need to be allowed to "probe" the system by sending repeated requests
6. Better schedules can be generated using the "flexible" approach. If high resource utilization is required, flexible requests can be relaxed to fit.

FUTURE WORK

The next step towards our goal of defining a potential Space Station/CDOS scheduling environment is to build a network of communicating ROSE schedulers in order to further refine distributed scheduling concepts.

REFERENCES

1. Ford Aerospace Corporation
"Integrated Resource Scheduling Final Report", 1987
2. GSFC Code 522, Data Systems Technology Division
"Interface Control Document for the Telescience
Implications on Ground Systems (TIGS) Joint Effort", 1988
3. Hansen, Sparn, and Davis; University of Colorado at
Boulder
"Concepts for Planning and Scheduling in the Space
Station Era", 1988
4. Hansen, Sparn, and Davis; University of Colorado at
Boulder
"Requirements for Space Station Telescience Command,
Control, and User Interface", 1986
5. GSFC Code 522, LASP, and FAC
"Telescience Implications on Ground Systems (TIGS)
Task Results Presentation" 1988
6. Leban, McDonald, and Forster,
"A Representation for Collections of Temporal Intervals"
AAAI-86

ACKNOWLEDGMENTS

We would like to thank Mike Tong (GSFC) for his support and guidance during this project. Additionally, the experience of the University of Colorado LASP personnel (Tom Sparn, Randy Davis, and Elaine Hansen) in satellite operations proved to be an invaluable aid.

Fault Isolation / Diagnosis

**MOORE: A Prototype Expert System for Diagnosing
Spacecraft Problems**

Achieving Real-Time Performance In FIESTA

**Mission Telemetry System Monitor: A Real-Time
Knowledge-Based System**

MOORE
A Prototype Expert System
for Diagnosing Spacecraft Problems

Katherine Howlin
Jerry Weissert, Kerry Krantz

Westinghouse Electric Corporation
7501 Forbes Boulevard, #104
Seabrook, Maryland 20706

ABSTRACT

MOORE is a rule-based, prototype expert system that assists in diagnosing operational Tracking and Data Relay Satellite (TDRS) problems. It is intended to assist spacecraft engineers at the TDRS ground terminal in troubleshooting problems that are not readily solved with routine procedures, and without expert counsel. An additional goal of the prototype system is to develop in-house expert system and knowledge engineering skills.

The prototype system diagnoses antenna pointing and earth pointing problems that may occur within the TDRS Attitude Control System (ACS). Plans include expansion to fault isolation of problems in the most critical subsystems of the TDRS spacecraft.

Long term benefits are anticipated with use of an expert system during future TDRS programs with increased mission support time, reduced problem solving time, and retained expert knowledge and experience.

Phase II of the project is intended to provide NASA (Code 405) the necessary expertise and capability to define requirements, evaluate proposals and monitor the development progress of a highly competent expert system for NASA's Tracking Data Relay Satellite. Phase II also envisions addressing two unexplored applications for expert systems, spacecraft Integration and Test (I&T) and support to launch activities.

The paper will discuss the concept, goals, domain, tools, knowledge acquisition, developmental approach, and design of the expert system. It will explain how NASA obtained the knowledge and capability to develop the system in-house without assistance from outside consultants. Future plans for a Phase II will also be presented.

1.0 INTRODUCTION

NASA/Goddard Space Flight Center, Code 405, Tracking and Data Relay Satellite (TDRS) Project developed a prototype diagnostic expert system to assess the feasibility of Artificial Intelligence (AI) technology in the form of an expert system for diagnosing on-orbit spacecraft problems. The proof of concept project was introduced in October 1986. Investigations of the technology, applications, and tools were initiated in January 1987 with attendance of the Knowledge Engineering Methodology Course by Teknowledge Inc. Knowledge acquisition and system design commenced in July 1987; the prototype and final reviews were complete in March 1988.

This paper describes the Phase I prototype expert system project. Section 2 defines the concept and objectives of the project. Section 3 describes the resources utilized to implement the system. Section 4 explains the capturing of the expert knowledge and Section 5 relates how the knowledge base was designed. Sections 3, 4 and 5 include lessons learned during development. Section 6 presents the recently approved future project plans.

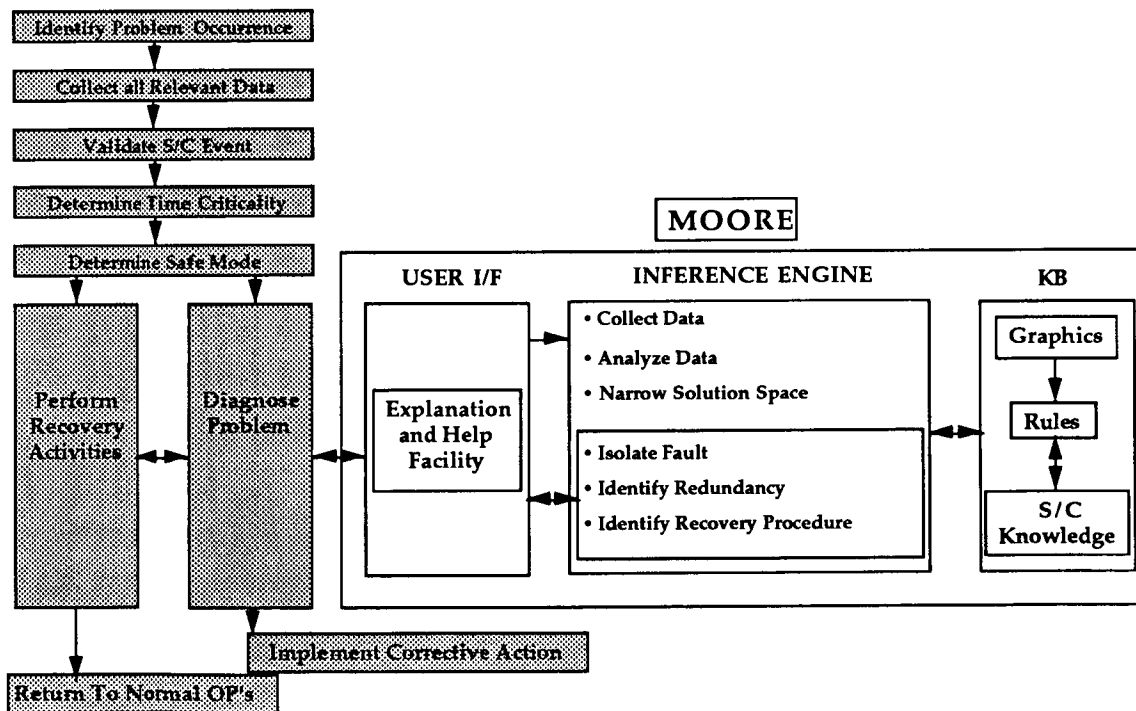


Figure 1 - The relationship of the prototype expert system to the current problem solving activities at the ground terminal.

2.0 CONCEPT AND OBJECTIVES

2.1 Phase I Concept

It is conceivable that an expert system can effectively be used to diagnose faults on an operational geosynchronous communications satellite. Use of such an expert system could reduce problem solving time, increase mission support time, and provide an ideal operator training tool. It was the intent of this prototype effort to prove that it is feasible and worthwhile to model an expert's knowledge and reasoning about a spacecraft subsystem and retain his experience utilizing expert system technology for use in solving on-orbit problems.

2.2 Phase I Objectives

There is reason to believe that contractors' proposals for the forthcoming Advanced Tracking and Data Relay Satellite (ATDRS) program will include an expert system(s); therefore, the NASA Tracking and Data Relay Satellite (TDRS) Project Office must have the capability to scope, define, evaluate and monitor anticipated expert system development and implementation. The objective of the prototype effort was to demonstrate how an expert system can be employed to diagnose problems in the scope of a finite domain and to gain the above designated capabilities. Furthermore, the objective included identifying a single, recognized expert for one of the TDRS subsystems (Attitude Control System [ACS]; Power; Telemetry, Tracking and Command [TT&C]; etc.) and capturing his reasoning, unique problem solving heuristics and thought processes using knowledge engineering techniques. Another objective was to determine whether a commercially available development tool may be tailored to the specific requirements of a diagnostic, spacecraft application.

The role of the prototype expert system MOORE is defined in Figure 1. The system was intended to support the spacecraft engineer in the diagnosis of the observed anomalous event and indirectly in the recovery activities, often executed in parallel with the diagnosis. MOORE was designed to operate in an environment where certain activities are performed prior to its involvement. These activities include validating that the identified problem is an actual spacecraft event rather than a ground station equipment, operator, or a command link problem. This function also includes validating the telemetry. The expert system requires valid spacecraft symptoms as input in order to reach meaningful conclusions. Placing the spacecraft in a safe mode configuration is another activity the off-line prototype does not attempt to perform. Even though the prototype is not responsible for performing these activities, the system does query the user about validation and safe mode status in order to ensure these functions have been addressed before proceeding with fault isolation and problem diagnosis.

3.0 RESOURCES

3.1 Selection of Domain, Expert and Development Tool

The domain, expert and development tool were selected in concert to ensure the success of the expert system development. The nature of the problem domain suggested specific features required of the development tool. Similarly, the expert was instrumental in better defining the scope of the problem domain.

Application of an expert system appeared to be more feasible and beneficial to some TDRS subsystems than to others. The TDRS Attitude Control System (ACS) was suitable because its complexity warrants expert system development, however, it is not as multifarious as the Payload or Telemetry, Tracking and Command (TT&C) subsystems. Furthermore, there was a history of well understood ACS problems which could be used to develop the expert system.

The selection of a qualified expert was more critical than the selection of a problem domain. Fortunately, the person with the necessary personality traits and highly regarded expertise, was the specialist for the ACS. Not initially obvious was the fact that it is rare for an identified expert to be a specialist in all aspects of a particular spacecraft subsystem. Mr. Robert J. Moore, of TRW, however, has been intimately involved with the design, assembly, integration, test and operations of the ACS.

The time of the expert, Mr. Moore, was hard to obtain. The specialist who is most in demand, therefore the least accessible, is the type of expert essential for the success of an expert system. Fortunately for the project, and surprisingly to most, Mr. Moore was willing to support the effort since he advocates a mechanism for retaining valuable expertise and experience of retiring or transferring engineers. The scope of the domain was then better focused based on preliminary knowledge acquisition discussions with Mr. Moore.

In addition to meeting criteria such as cost, availability, ease of assimilation, speed, size, and vendor support, the tool selected for the project had to complement the problem domain characteristics. The characteristics of the domain specify the required adequacies in the following areas: inference strategy (forward or backward chaining or a combination); a rule-based, frame-based or object-oriented scheme for representing knowledge; ability to handle uncertainty and use of certainty factors; knowledge base size; and graphics interface.

Since ACS problem resolution is goal oriented, a backward chaining inference strategy was desirable. However, a forward chaining capability was also desirable if available in the same tool. Uncertainty was anticipated in the domain, therefore, the

tool had to be able to handle various levels of confidence in problem evidence. The system required explanation facilities to support varying levels of users, especially since it was built as a demonstration expert system for a diverse audience.

Since the intent was to characterize an expert's thought processes and heuristics, a rule-based system, where knowledge is stored in the form of if-then rules, was most suitable for representing ACS facts and relationships. Object-oriented or frame-based tools could have been effectively employed, but were not absolutely necessary given the funding resources available.

3.2 Tool Comparison and Recommendations

The tools under consideration were tested by building a small prototype. M.1 by Teknowledge and Personal Consultant Plus (PC Plus) by Texas Instruments were tested by building rules similar to those anticipated for the expert system. The final decision to use PC Plus resulted from this informative exercise. PC Plus was selected over M.1 because it provided:

- o Smoother graphics integration,
- o Superior editing capability,
- o Preferable development environment, and
- o Attractive local support and documentation.

Furthermore, the PC Plus development package was available for less than half the cost of the comparable M.1 package.

In addition, the areas in which M.1 was stronger than PC Plus were not meaningful to this project. Teknowledge emphasized M.1's greater speed and ease of integration with external databases and programs. The off-line prototype system did not require maximum speed; likewise, it was not a prototype objective to interface with existing software applications.

It was desirable for the development tool to be well suited to the computer skill level of the domain expert. The PC Plus software provided English translations of the SCHEME (a simple, modern version of LISP) rules. Also, there were only two function keys to remember during consultation; all other user interaction is menu-driven or directed with a prompt.

3.3 Hardware and Graphics Selection

For prototype development, PC Plus was installed on an IBM AT compatible with expanded memory, enhanced color graphics board and an 80386 microprocessor. The 80386 microprocessor provided a significant increase in speed over the 80286 version.

The final selection was the graphics package. It was difficult to find a package in-house which was compatible with both the hardware and the software. Of the many examined, FREELANCE by

LOTUS Development, Inc. was the only graphics package which interfaced satisfactorily with PC Plus and the selected hardware. It was later evident during demonstrations how critical graphics were to ultimate system acceptance; the persistent search was justified. Once FREELANCE was selected, integrating the created graphics into the knowledge base was straightforward with a compression routine and a simple function call.

4.0 KNOWLEDGE ACQUISITION

The most challenging and time consuming component of constructing the expert system was obtaining the knowledge from the expert. Discussed in the following four sections are the steps undertaken to achieve this goal.

4.1 Preparation

Documented preparation techniques for the knowledge acquisition process were applied effectively. For example, prior to the first interview, the knowledge engineers familiarized themselves with the TDRS Attitude Control System. Reading appropriate specifications, documents and schematics enabled the knowledge engineers to better communicate with the expert. This familiarization with the expert's domain prevents unnecessary interruptions by the knowledge engineers as the expert recounts his thought processes. In order to more clearly understand how the knowledge acquisition process transpires, an auto mechanic was interviewed with respect to another diagnostic domain. Based on that session, the knowledge was organized into a 20 rule prototype system. This preliminary groundwork provided for an overall approach and better prepared the knowledge engineers.

4.2 Motivation

The enthusiasm of the knowledge engineers was essential. If the expert perceived a true sense of interest in his subject, he was complimented and readily motivated. Rapport building approaches suggested by experienced knowledge engineers proved effective, such as using the expert's name in the title of the system. Fortunately, peaking the expert's interest was not the greatest challenge. Mr. Moore, approaching retirement, was more than interested in retaining his expertise for posterity. However, in the early stages of knowledge acquisition it became apparent that the expert had his own idea of how to capture his problem solving strategies, which was not compatible with expert system techniques. This presented a problem, that to our knowledge, had not yet been encountered; no suggested solutions were readily available. To steer him in the proper direction, a demonstration of a diagnostic expert system was beneficial. This provided Mr. Moore a clearer understanding of the system to be produced and exactly what type of information was required of him.

4.3 Initial Knowledge Acquisition

Initial knowledge acquisition interviews provided the knowledge engineers a foundation for building an understanding of the problem domain and the expert's recognized problem solving approach. Furthermore, the preliminary questions provided a skeletal structure of the entire anomaly resolution process. Questions such as how the current problem solving practice evolves and when the expert's contribution is required were asked. The expert was also queried with respect to how he categorizes problems within the ACS domain. The knowledge engineers learned, in the initial phases of knowledge acquisition, what resources the expert utilizes in solving problems such as TDRS documentation, schematics component failure histories and respected colleagues.

4.4 Knowledge Acquisition Approach

The approach to extracting knowledge was case-directed. Actual spacecraft events documented in the Spacecraft Orbital Anomaly Reports (SOAR) in which the expert was an integral part in solving, were used as a basis for knowledge acquisition. It was easier for the expert to recall an actual thought process rather than reason about a set of hypothetical conditions. There were times when the expert, lacking confidence in his response, consulted with reference material or a colleague. This was done outside the interviewing session since it was not proposed to deal with multiple expert input.

Trying to extract every step of the thought process the expert used to arrive at a conclusion was difficult due to the expert's natural heuristic leaps. A list of key phrases was effectual in uncovering those steps that the expert skips over due to his vast experience. Specific questions such as "Try to recount aloud how you proceeded and recall each discrete thought that went through your head at the time you were solving this anomaly" and "What is common knowledge; would anyone else know this?" were used to guide and focus the expert on crucial problem solving steps. All knowledge acquisition sessions were taped for the benefit of the knowledge engineers and for reference during knowledge base reviews with the expert.

5.0 DESIGN AND DEVELOPMENT

5.1 Methodology

The design methodology utilized was based on the model of the development phases for building expert systems in "A Guide to Expert Systems" by Donald A. Waterman. A domain consisting of the TDRS ACS is obviously far too comprehensive for a prototype effort. Therefore, based on initial knowledge acquisition in the reformulation cycle, the domain was narrowed to a more manageable subset of the ACS. With the domain better defined, the knowledge

engineers extracted specific knowledge which was then organized and formed into rules. These rules were then validated by the expert which generated two iterative cycles. In the redesign cycle, the knowledge base was expanded with new knowledge (i.e., expanded in breadth). In the refine cycle, the knowledge previously obtained was made more accurate (i.e., expanded in depth). In the review cycles, the expert consulted the system to verify content; he was not trying to understand how the coded rules were generated. All cycles were repeated through many weeks of interviews extending over a seven month period. Approximately 72 hours of the expert's time was used.

Knowledge acquisition experience suggested that the optimum conditions for building a comprehensive expert system include knowledge engineers working full time, collocated with the expert. Long periods between knowledge engineering sessions were detrimental to development due to the complexity of the subject. The technical details were quickly lost. Both the expert and knowledge engineers continually needed to be refreshed.

The refine cycle was also executed with several project managers and members of the AI community. It was important to obtain key managers input in the design process so that when the proof of concept was challenged, substantial support had already been secured.

5.2 System Functions

As a result of this development process, MOORE was constructed, as depicted in Figure 1. The system operates by querying the user for the spacecraft problem symptoms. The relevant data is collected, analyzed and via forward chaining rules, leaps of inference are made that provide intermediate conclusions. These intermediate conclusions explain to the user how the system is proceeding and why it has identified or eliminated certain components. The solution space is narrowed, primarily operating on a backward chaining mechanism until the most probable fault is isolated. The system then identifies redundancy and recovery procedures when necessary. In conjunction with the inference engine, the knowledge base provides the essential rules and specific spacecraft knowledge needed to deduce conclusions. Graphics, created with FREELANCE by LOTUS Development Corporation, are accessed to provide the user additional technical spacecraft information. At any time during the consultation, the user can question the reasoning of the expert system or obtain additional information through the explanation and help facility.

5.3 System Architecture

The MOORE knowledge base includes approximately 100 rules. The rules were created in the Abbreviated Rule Language (ARL) or in SCHEME (TI's form of LISP) in the format,

IF: Combination of CONDITIONS THEN: CONCLUSIONS AND ACTIONS

```

If  1) the corresponding GDA reference data is as expected in the telemetry,
    and
    2) null widths are shorter than expected, and
    3) the signature of the null telemetry is Indication Worked
    Intermittently Before Complete Loss Of Null,
Then 1) Inform the user of this decision, and
    2) display a graphic picture, and
    3) display a graphic picture, and
    4) it is definite (100%) that the null switch is not operating properly.

IF: GDA-TLM AND SHORT-NULL-WIDTHS AND NULL-SIGNATURE = "Indication Worked
    Intermittently Before Complete Loss Of Null"
THEN: PRINT :ATTR QUOTE RED "The system has determined that there is a Null
    Switch problem" :ATTR QUOTE WHITE :LINE 2 "The shorter than normal null
    widths imply that the switch may have been closing late and/or opening
    early indicating degradation of the switch. Since, the null switch is
    an electromechanical device it is possible that it could fail
    intermittently in contrast to the electronic devices that would not fail
    intermittently. Therefore, there is a strong possibility that the null
    switch has failed." :LINE 12 AND PICTURE "SWITCH" AND PICTURE "NULLDETE"
    AND NULL-SWITCH-FAILURE

PREMISE: ($AND
    (SAME FRAME GDA-TLM)
    (SAME FRAME SHORT-NULL-WIDTHS)
    (SAME FRAME NULL-SIGNATURE "Indication Worked Intermittently
    Before Complete Loss Of Null"))
ACTION: (DO-ALL
    (MPRINTT :ATTR
    (QUOTE
    (RED)) "The system has determined that there is a Null Switch
    problem" :ATTR
    (QUOTE
    (WHITE)) :LINE 2 "The shorter than normal null widths imply
    that the switch may have been closing late and/or opening early
    indicating degradation of the switch. Since, the null switch is an
    electromechanical device it is possible that it could fail
    intermittently in contrast to the electronic devices that would not
    fail intermittently. Therefore, there is a strong possibility that
    the null switch has failed." :LINE 12)
    (PICTURE "SWITCH")
    (PICTURE "NULLDETE")
    (CONCLUDE FRAME NULL-SWITCH-FAILURE YES TALLY 100))

UTILITY: 99
  
```

Figure 2. Rule 76 represented in three formats: English translation, Abbreviated Rule Language (ARL), SCHEME.

Specific spacecraft knowledge, facts and relationships were stored within parameters used by the rules as well as in the rules. The rules were grouped into what Texas Instruments refers to as "frames" so that the inference engine efficiently accesses a limited subset of rules in any diagnosis, as opposed to exhaustively searching through all rules. An example of a rule (RULE 76) represented in English, ARL, and in SCHEME, respectively, is provided in Figure 2.

Rule 76 fires, or is executed, during a consultation only when all three conditions in the premise become true. The first condition, or value of the parameter, GDA-TLM, is determined by another rule (RULE 111). In order for Rule 111 to fire, input is requested from the user with a graphic prompt, created by the developer, represented in Figure 3.

The value of the second parameter, SHORT-NULL-WIDTHS, in the premise conditions is determined by directly prompting the user as depicted in Figure 4. Since only a TDRS ACS expert can determine whether a particular null width is "short" or not, HELP is available to the user through the help facility by selecting the F1 function key. The graphic help associated with the prompt is presented in Figure 5. The value of the third parameter traced is determined with a prompt similar to that in Figure 4.

The System is Collecting Data

Indicate which null reference (or combination of references) is not being obtained during the null search procedure?







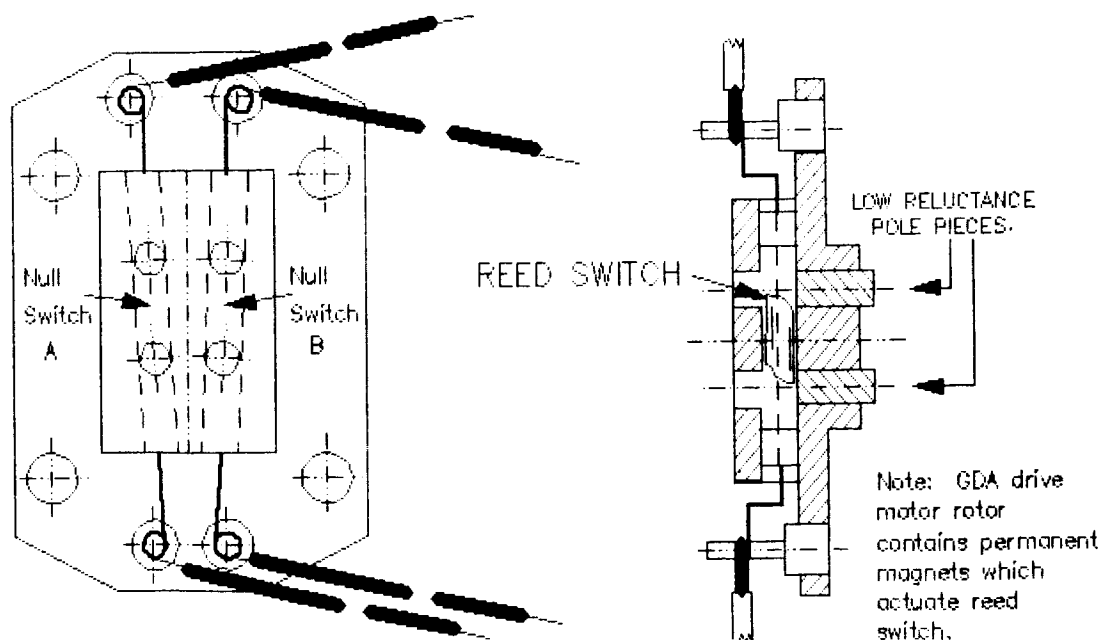
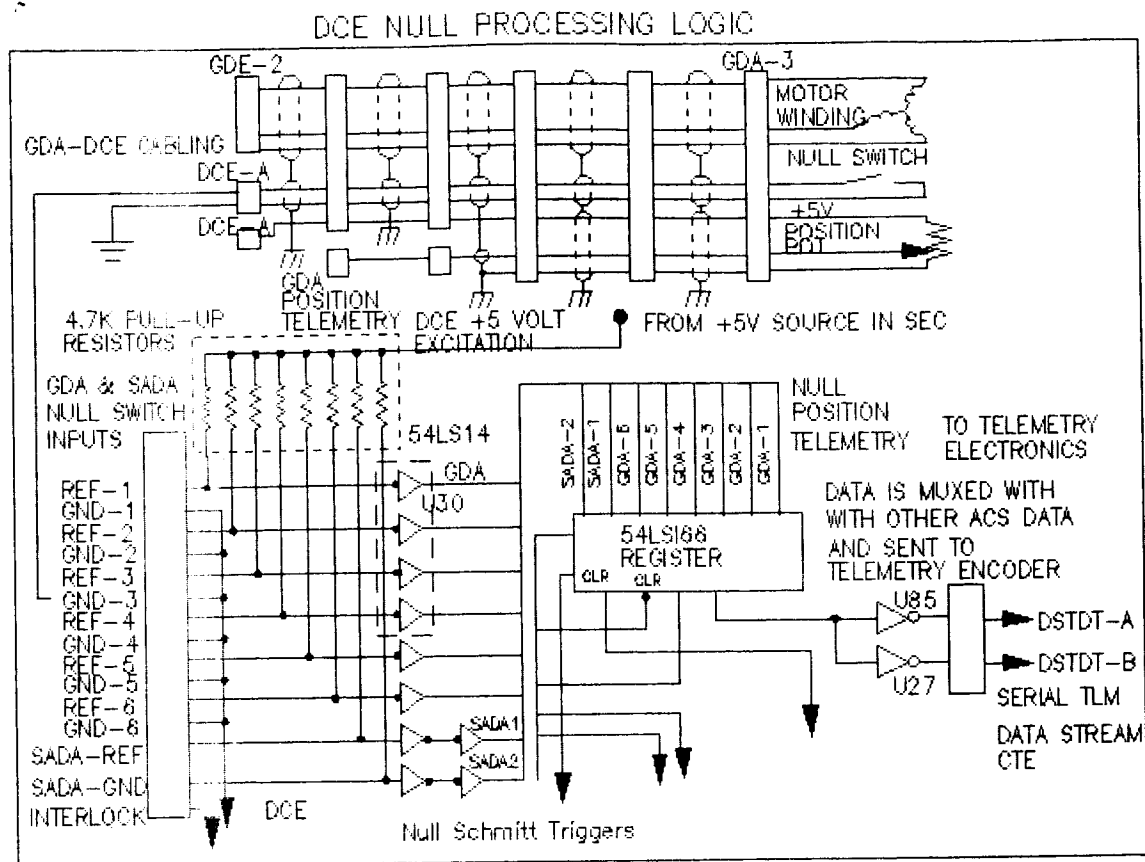
- | | | | |
|----|--|----------|--|
| 1. | GDA1 | NULL REF |  |
| 2. | GDA2 | NULL REF |  |
| 3. | GDA3 | NULL REF |  |
| 4. | GDA4 | NULL REF |  |
| 5. | GDA5 | NULL REF |  |
| 6. | GDA6 | NULL REF |  |
| 7. | "Problems obtaining several/all null references" | | |

Figure 3. Example of a graphic prompt to the system user. The user selects the appropriate number.

ORIGINAL PAGE IS
OF POOR QUALITY



If all three conditions become true during diagnosis, Rule 76 fires and performs four actions. As a result of the first action, text is printed stating the conclusion. Two explanatory graphics are displayed as represented in Figures 6 and 7. The last action concludes NULL-SWITCH-FAILURE with a certainty factor of 100. NULL-SWITCH-FAILURE was one of the intermediate goals or subgoals of the expert system diagnosis.

When a final diagnosis is made, the system identifies the appropriate redundant configuration. Figure 8 delineates the redundant configuration to be used in the case of a null switch failure.

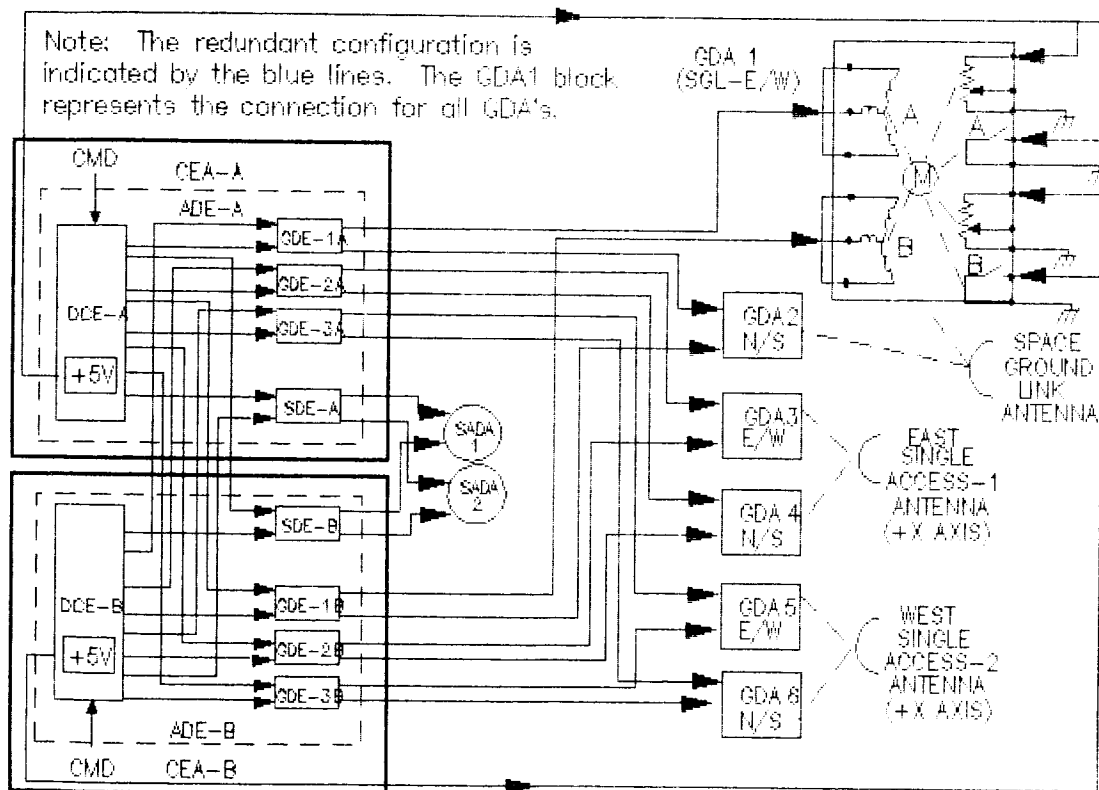


Figure 8. Identification of the recommended redundant configuration based on an identified fault. Appropriate connections are highlighted in blue on the color monitor.

The areas of the TDRS ACS currently characterized in the prototype, MOORE, are depicted in Figure 9. The problem domain is shaded; problem sources are identified in black. As evident in the diagram, the system can identify one of many failures as the cause of the spacecraft problem given a set of symptoms.

ORIGINAL PAGE 1,
OF POOR QUALITY

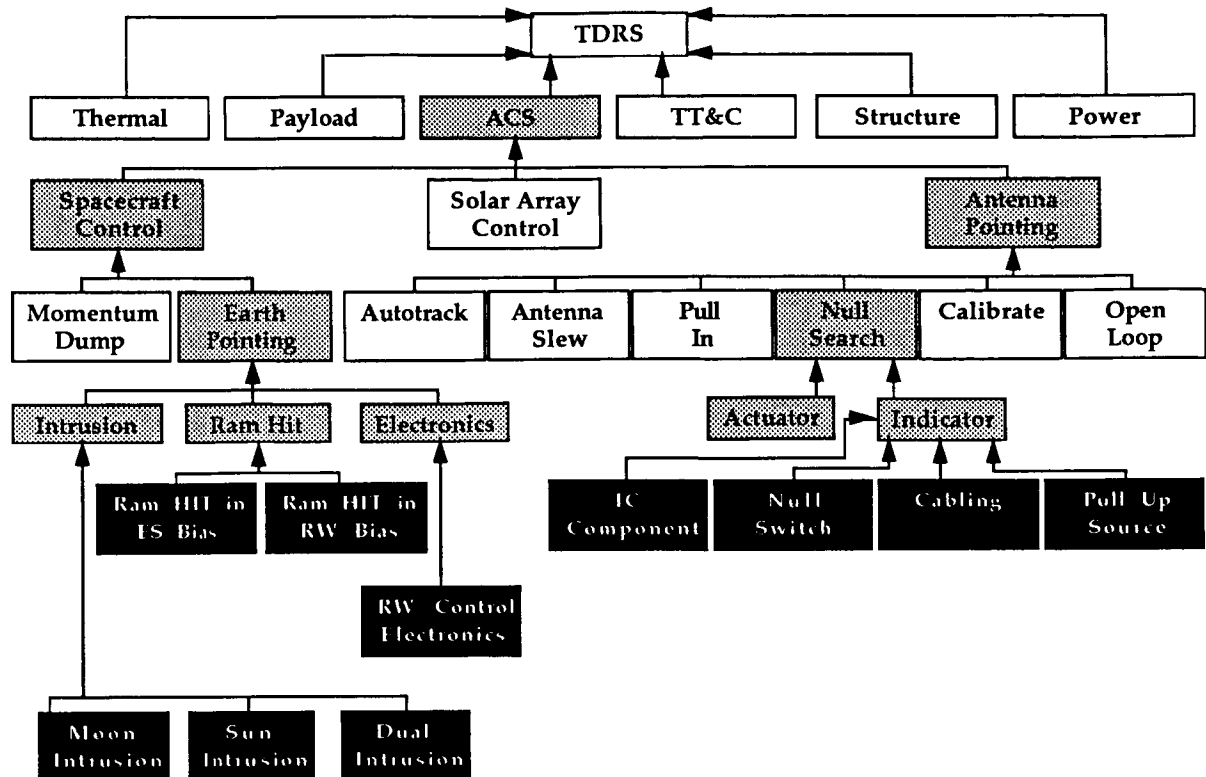


Figure 9. Scope of the problem domain for the prototype expert system, MOORE. Areas currently characterized are shaded.

6.0 FUTURE DIRECTIONS

6.1 Phase II Purpose

Phase I, development of the prototype expert system (MOORE), was implemented to prove that a diagnostic expert system was feasible and had a beneficial application to the TDRS program. Upon successful demonstration of the expert system capabilities, Phase II was approved. The objectives for Phase II are currently being finalized.

The ultimate goal is to provide NASA, Code 405, an expert system which can be utilized for TDRS fault isolation and problem diagnosis. It is expected that such a system will be a contract deliverable with the next generation of TDRS spacecraft. Phase II is intended to provide Code 405 the necessary expertise and capability to define requirements, evaluate proposals and monitor the development progress of a highly competent diagnostic expert system for NASA's Tracking Data Relay Satellite.

6.2 Phase II Objectives

The expertise will be developed by studying, implementing and evaluating the more complex knowledge engineering and expert system tasks that were beyond the scope of Phase I. The following are targeted objectives for Phase II:

1. **Interview potential users to optimize operator interface.** Elicited user feedback will be utilized to make the interface efficient and user friendly. Special attention will be given to effectiveness and standardization of colors, symbols and prompts. Targeting the system to the technical level of the most frequent user is most crucial to user acceptance of the system.
2. **Interfuse knowledge of multiple experts.** Due to the lack of availability of a particular expert for each subsystem, it becomes necessary to interview more than one for a given subsystem. Conflicting information and differences in problem solving rational must be handled with no detriment to the system. Schemes for dealing with this will be explored and evaluated.
3. **Assess the applicability of expert system technology to the spacecraft Integration and Test (I&T) program.** I&T problems are often of a different nature than those experienced on orbit. Examples of these type problems are operator errors, software problems, and cabling and test equipment errors. Additionally, problems discovered in testing are similar to on-orbit anomalies.
4. **Accessing external data bases.** The spacecraft manufacturer maintains a data base of all problems discovered at box and higher level testing. All on-orbit problems are catalogued. Additionally, there is a GSFC data base of all problems observed on Goddard managed spacecraft. A method will be devised to access this type of information in existing formats.
5. **Evaluate a Real Time Telemetry Interface.** The transition from a passive (off-line) to an active (on-line) system that would accept a real-time telemetry input, evaluate spacecraft status, and make recommendations to the operators for commands and corrective action is a major step. The degree of sophistication and operator confidence required to implement this approach will require considerable investigation and evaluation. This is an important direction to explore in a step towards making spacecraft autonomy more realizable.
6. **Evaluate application to launch activities.** Utilizing the expert system as a training tool for the flight support team and a diagnostic tool during simulation and launch will be investigated.

7. **Expand cognizance of available tools and technology.** With Artificial Intelligence software and expert system technology evolving so rapidly, application and implementation options must be continually reviewed.

The scope of the problem domain for accomplishment of Phase II objective will either be an expansion of the prototype to include a more comprehensive subset of the ACS, or it will include a second TDRS subsystem selected by management for its criticality to the TDRS mission and the existence of viable experts. In either case, it is understood that the knowledge characterized by the prototype will be incorporated into the Phase II system.

Having completed the Phase II tasks, the project should have the experience and expertise to understand, define and scope the task, ask technical questions from a position of insight, and assess proposals. Also, as a result of the Phase II activities, the project should be capable of adequately monitoring the selected contractor's progress in development and implementation of a comprehensive expert system.

ACKNOWLEDGMENTS

Westinghouse Electric Corporation engineers developed the prototype expert system, MOORE, for and with guidance by the NASA TDRS Project Office, Code 405. The authors would like to acknowledge the following indispensable contributions:

Mr. Laurence Goodman, Project Support Manager, conceived and managed the project and its development.

Mr. Richard Meyers conceived the application based on his study of AI technology.

Mr. Robert J. Moore elucidated on his design problem solving experiences and was instrumental in designing the consultation format.

Additionally, Mr. Thomas Williams' support was very decisive in establishing the concept; furthermore, he provided insightful input based on his experience as TDRS Systems Manager.

Achieving Real-Time Performance in FIESTA

William Wilkinson, Nadine Happell, Steve Miksell and Robert Quillin
Stanford Telecommunications, Inc.*†

Candace Carlisle‡
NASA/GSFC

Abstract

The Fault Isolation Expert System for TDRSS Applications (FIESTA) is targeted for operation in a real-time online environment. Initial stages of the prototype development concentrated on acquisition and representation of the knowledge necessary to isolate faults in the TDRSS Network. This paper describes recent efforts focused on achieving real-time performance including: a discussion of the meaning of FIESTA real-time requirements, determination of performance levels (benchmarking) and techniques for optimization. Optimization techniques presented include redesign of critical relations, filtering of redundant data and optimization of patterns used in rules. Results are summarized.

1 Introduction

The Fault Isolation Expert System for TDRSS Application (FIESTA) is an operator decision aid targeted for deployment in NASA/GSFC's

Network Control Center (NCC). FIESTA is intended to assist operators who isolate and diagnose faults in the Space Network. Operation is to be continuous. The automated inputs to FIESTA are a stream of network control and status messages. This paper covers our efforts to prepare FIESTA for keeping up with that message stream in real time.

Section 2 is an overview of the FIESTA environment. [lowe87] is a more comprehensive introduction that covers operational issues and some architectural details. Section 3 discusses FIESTA's real-time performance requirements. Optimization techniques are covered in Section 4; the implementation plan is sketched in Section 5. Sections 6 and 7 present results and a brief conclusion.

2 Background/Overview

FIESTA's domain focuses on fault detection and diagnosis of NASA's Space Network (SN). The SN combines space and ground elements to provide tracking and data relay services for spacecraft in near-earth orbit. The space segment of the SN baseline is currently one operational geostationary Tracking and Data Relay Satellite (TDRS) which will grow to a multi-satellite constellation over the next five years. Through a systematic program of satellite replenishment and ground refurbishment,

*1761 Business Center Dr., Suite 400, Reston, VA 22090.

†FIESTA development has been undertaken by Stanford Telecommunications, Inc. as a subcontractor to Computer Sciences Corp. under NASA contract NAS5-31500.

‡Code 532.3, NASA/Goddard Space Flight Center, Greenbelt, MD 20771.

the SN is expected to support programs such as the space station and space telescope well into the next century.

User spacecraft telemetry and commands are relayed through Tracking and Data Relay Satellites (TDRSs) and downlinked to the White Sands Ground Terminal (WSGT) in New Mexico. Collocated with the WSGT is the NASA Ground Terminal (NGT) which provides communications interfaces for transferring data from WSGT to the other SN elements and users, via the NASA Communications Network (NASCOM). The Network Control Center (NCC) is the operational control facility responsible for managing this geographically distributed network of elements. Primary NCC functions include resource scheduling, equipment configuration direction, and service quality monitoring and assurance. The NCC coordinates all SN problem resolution. Figure 1 is a network overview.

Based on requests from user spacecraft control facilities, the NCC schedules *events*, consisting of one or more communications services for a single user for a single pass by a TDRS. During support of these events, performance and status data from network elements are transmitted to the NCC where the contents of these messages are combined and presented on display screens. NCC operators known as Controllers and Performance Analysts monitor these displays, as well as ground control (e.g., reconfiguration) messages that alter scheduled equipment configurations or effect operational changes to detect problems and determine appropriate courses of action.

FIESTA's purpose is to provide an intelligent assistant to network operators that will continuously monitor selected communication services. FIESTA detects and notifies controllers of faults, isolates problem sources to major system component levels (e.g., WSGT) and recommends resolution strategies. In order to adapt to the dynamic nature of the op-

erational environment, FIESTA was designed to recognize the various states of a user service and detect transitions from state to state (Figure 2). The three primary states for a user are *acquiring*, *nominal*, and *anomalous*. FIESTA's knowledge base is partitioned into context-limited rule sets applicable only in the appropriate service state.

FIESTA reasoning must first determine positive signal acquisition to identify the transition from acquiring to nominal. The system then monitors the service and detects variances from expected behavior, recognizing nominal to anomalous transitions. Transition into an anomalous state opens a diagnostic episode, in which FIESTA verifies the fault condition, hypothesizes potential fault locations and fault causes, and ranks these hypotheses based on rule interaction [miks87]. Once a diagnostic episode has been resolved, the system determines that a transition from *anomalous* to *nominal* has occurred.

Control and Status Messages

FIESTA's primary source of information about the situation of the Space Network is a series of control and status messages called "High Speed Messages" or HSMs. The four types of HSMs currently used in FIESTA are:

SHO Scheduling Orders allocate resources to given users.

OPM Operations Messages notify operators of various events or request/acknowledge various actions.

ODM Operations Data Messages are reports on the health and status of every ongoing service every 5 seconds. Contents vary by type of service. ODMs are the primary source of data for detecting faults. They are used for the example of Section 4.

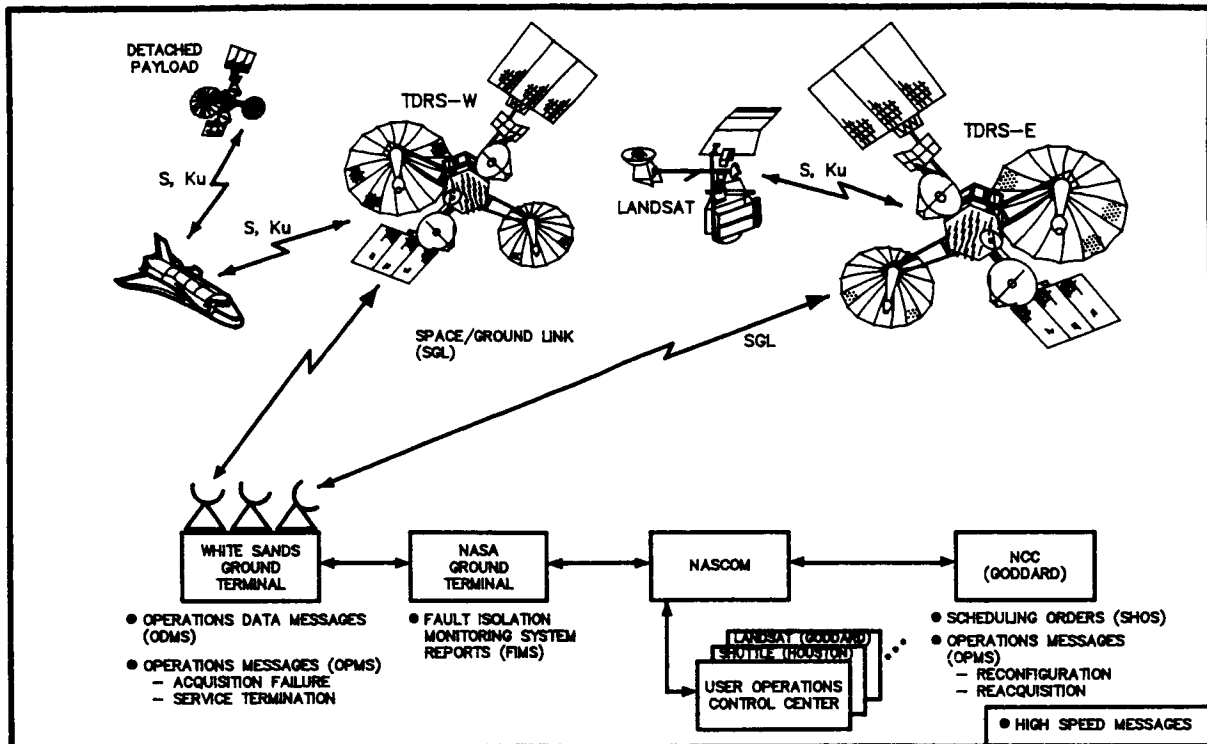


FIGURE 1: SPACE NETWORK OVERVIEW

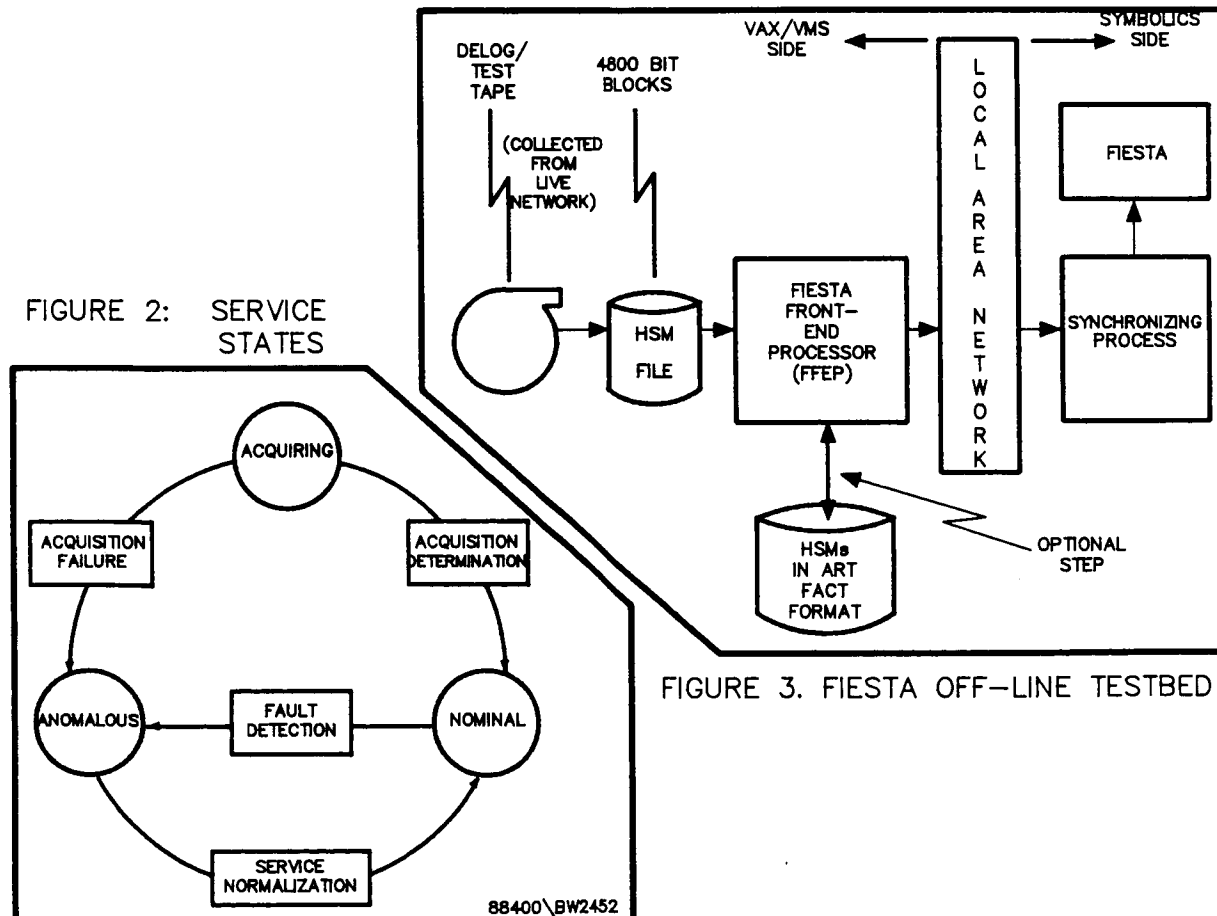


FIGURE 2: SERVICE STATES

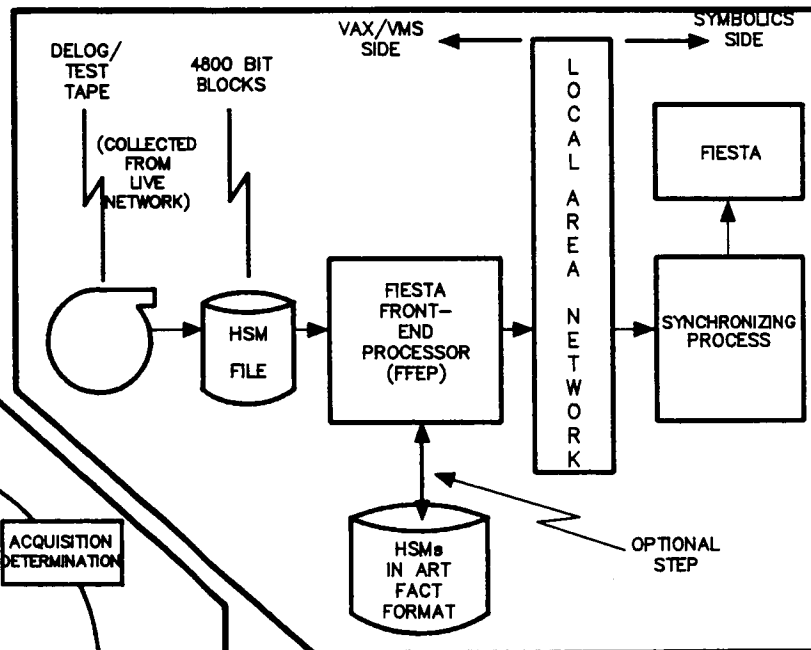


FIGURE 3. FIESTA OFF-LINE TESTBED

FIMS Fault Isolation Monitoring System messages contain data reported by monitoring equipment called "frame analyzers." FIMS reports are sent every 5 seconds for each ongoing service that is being analyzed (an option at NGT).

FIESTA views Network function in terms of services. SHOs are organized by events. The major organization of ODMs and FIMS messages is in terms of TDRS satellites and bandwidths. OPMs are organized by service.

The Off-Line Environment

FIESTA is currently a prototype that resides in STI's software development laboratory. For our development, testing and benchmarking we use copies of actual HSMs from the Space Network. In addition, network-wide simulations called "ESTL tests" provide additional data. The off-line suite (Figure 3) consists of a VAX/VMS system hosting the FIESTA Front End Processor (FFEP) connected to a Symbolics Lisp machine hosting the expert system. FIESTA was built using the Automated Reasoning Tool (ART) from Inference Corp.

The FFEP performs two major functions. It translates HSMs into s-expressions¹ for use by the expert system and it feeds them to the expert system at a controlled rate. The rates used most often are 1) as fast as possible and 2) driven by the encoded wall clock time of the original messages (or limited by available resources). FIESTA is designed so that the Lisp machine software is oblivious to the fact that it is in an off-line testbed. With the possible exception of a timer to detect message dropouts, we anticipate no major design changes in the on-line expert system software. We currently lack a real-time timer on the Lisp machine side; the development process often

requires that we stop and look at the current process or that we back up and resubmit situation data to the expert system after software modifications. These stoppages would appear as message dropouts to a system with a timer.

3 Real-Time Requirements

We are preparing to move the FIESTA prototype out of the laboratory and connect it to live data in the NCC. This motivates our consideration of real time.

Real-time in the software realm is many things to many people. Fuzzy definitions abound, many having to do with software interacting with physical items external to the host computer system and characterized by unpredictability. In some sense, all interactive software is real-time in that it synchronizes with unpredictable external devices (humans) via input mechanisms.

A fundamental problem now being addressed in real-time AI research is that most AI problems involve searches and other computations which are often NP complete [norm85, orei85], i.e., the time taken to complete a calculation is a nondeterministic polynomial function. The amount of computation required to solve them often cannot be computed a priori and sometimes cannot even be bounded. The state-of-the-art for such problems is often to build something and see if it works in real time. Analysis to guarantee required responses is frequently lacking. The analytical foundation to do so in general does not exist. Performance problems are solved via the bigger hammer theory: if it isn't fast enough, buy a faster machine. Software engineering tools are needed to answer questions such as:

- How can response times be calculated or bounded, especially when required re-

¹symbolic expressions, in our case using Lisp list syntax.

sponse time is not known a priori?

- When distributing systems, how does one decide when it is no longer profitable to divide and conquer a problem because of communications or other overhead?
- How can real-time knowledge-based systems be verified/validated?

Currently research is heavily weighted in favor of examination of ways to make systems faster over ways to insure that they are fast enough.

O'Reilly and Cromarty [orei85] present "a simple but nonetheless formal definition of real-time performance" that does quite well for our purposes. They claim that real-time systems are absolutely required to provide outputs by some deadline time. They also suggest ways in which systems can monitor progress towards solutions and take appropriate actions as deadlines approach. [less88]'s approach involves choosing approximation techniques based upon solution time available. [laff88] surveys the state-of-the-art of real-time KBS but does not dwell on O'Reilly and Cromarty's definition in its survey of existing systems. We shall do the same.

FIESTA is currently subject to loose real-time requirements such as "inferencing delay shall not exceed 3 minutes" and "determination that a fault exists shall be made within one (1) minute of receipt of data (by the expert system component) which indicates a possible anomaly . . . Results of the initial diagnosis shall be available with [sic] one minute of fault detection" [sti87a]. What these requirements say is that the FIESTA system is allowed to lag real-time data by some maximum period. Average lags in processing of situation data² should be much smaller than maximum

allowed lag.

FIESTA requirements seemingly mandate a hard real-time architecture. Deadlines are required to be met. But the actual architecture used in FIESTA has no notion of these requirements built into it. Instead, every unit of situation data is processed to completion before new data is accepted. There are no deadline mechanisms. We call this a "soft" real-time system. Using new data before old data is processed to completion or discarded would require significant redesign; we are not even sure about what such an architecture would look like or if we would be able to achieve diagnostic results of comparable quality in a hard real-time FIESTA.

The soft real-time approach has achieved acceptable performance for the heaviest loadings we have been able to find in the taped HSMs available to us. The broadest measure of real-time performance is the maximum lag in processing of real-time data inputs when they are fed to our expert system host at wall clock rates. Maximum overall lag is bounded by on-line testbed requirements at 3 minutes. Worst case lags observed in our laboratory using realistic data collected from the live network are on the order of several seconds. True worst case input data sets have not been constructed. These would involve 2 TDRSs with the maximum possible number of services starting/stopping/ongoing and realizing some combination of faults simultaneously. This is a much higher system loading than is anticipated for the TDRSS network. It is also a much higher loading than the on-line testbed is required to handle: two simultaneous events, each event consisting of some limited number of services. Typically Shuttle events have three services, non-Shuttle events have two.

After FIESTA initialization, the greatest lags generally occur at two times: service startup time and during fault detection and

²We refer to *situation* data as distinct from *control* data like operator commands (which have their own real-time requirements).

diagnosis. These may be characterized as the times when situation data is changing the most. Steady state performance has been somewhat faster than real time for typical data sets. Running in the "as fast as possible" mode, the off-line testbed is usually limited by the FFEP host, a venerable but low-powered VAX 11/730. ART uses a modified Rete algorithm. The Rete algorithm [forg80] is optimized for data that changes infrequently.

ODMs (a type of status message) are sent every 5 seconds, on average, for every ongoing service. FIMS reports are also grouped by TDRS and are on a similar 5 second cycle but are limited to services monitored by frame analyzers at NGT. Naturally, when faults occur, situation data is changing. In the worst case (from the Rete algorithm perspective), data changes with every "data cycle" or with every new message. Many HSM values are binary; some can flip-flop in fault situations.

4 Speeding Up FIESTA

The initial FIESTA design philosophy was that timing should be for the worst case: every HSM fact³ changing in every data cycle⁴. This is an attractive goal; it would effectively make FIESTA more like a hard real-time system with a 5 second bound for processing of each HSM. Actually, messages arrive more fre-

³Some ART terminology: a *relation* is a pattern for a fundamental unit of information in an ART database, including a symbol that defines the relation. A *fact* is an instance of a relation—a filled pattern that is part of the database. Facts also include context which is unimportant in this discussion.

⁴A better bound could be generated from the arrival statistics of HSM fragments given the limited speed at which blocks of HSMs are transmitted and the maximum size of HSMs. HSMs consist of 1 to 15 blocks of 4800 bits each. HSM blocks are in general interleaved with other messages and assembled into whole messages by receivers. The NASCOM data rate is 56 KBPS in and out of the NCC.

quently than every 5 seconds because of overlaps in data cycles. Overlaps typically occur three ways: 1) ODMs and FIMS reports for a TDRS are not synchronized, 2) when multiple TDRSs are active, their reports are not synchronized and 3) miscellaneous messages (OPMs and SHOs) are mixed into the HSM stream at unpredictable times. This means that the bounded time would have to be somewhat less than 5 seconds to process an entire HSM before the next one arrives. Alternatively, processing 5 seconds worth of messages in 5 seconds would work as well.

Results with this architecture were disappointing. We were not able to keep up with even nominal loads in our laboratory setup. This caused us to rethink our worst case timing strategy and embrace the idea that it really is acceptable to fall behind in processing upon occasion. The corollary is that the system must be able to 1) buffer messages to avoid losing data and 2) process those buffered messages faster than the wall clock time in which they were generated, in order to catch up to the current data stream. The danger of our new philosophy is that the system is most likely to lag real time when it is most needed, during fault situations when it has to do its most "thinking." We believe that the performance we have achieved is more than acceptable. It is well within the required bounds and seems to be comparable to the speed of the human operators FIESTA is meant to assist. Note that multiple simultaneous faults are relatively rare; since less processing power is now required to monitor healthy services, more processing power can be focused on faults that occur while other services are ongoing.

The "bog down and catch up later" design provides additional benefits. More processing power is available for housekeeping chores during nominal operation, particularly garbage collection. Response times for the operator interface are also improved. The fact that we

can process nominal data quickly also speeds development/demonstrations (and training!) because we can speed through the less interesting portions of input data sets.

The rest of this section concentrates on what we did to speed up FIESTA's expert system. Results are presented in Section 6.

Situation Relation Redesign

FIESTA is primarily rule-based. The relations that represent HSM data are used in many of the rules. We realized our greatest efficiency gains by redesigning HSM relations. Several possible redesigns were considered. The most productive redesign idea was a combination of two ideas: 1) redesigning HSM relations to use shorter, more specific patterns and 2) filtering redundant HSM elements. As we shall see, filtering was made possible by redesign. The motivation for smaller relations is that shorter, more specific relations are generally matched fewer places than larger relations; thus the cost of changing them via retractions and reassertions is lower. The motivation for filtering is that ODMs and FIMS reports are typically highly redundant. They often change only a time stamp⁵ from message to message for a given service. The unnecessary processing that redundant data causes is explained later.

This discussion will focus on ODM relations because they are more crucial to performance than other HSM relations simply because of their frequency; they are in the HSM stream for every ongoing service, every 5 seconds. This discussion is applicable to other relations, particularly FIMS reports, which are also sent frequently when used.

One ODM message reports upon all ongoing services for a TDRS. The original FFEP trans-

lated an ODM into one ODM fact⁶ with a subset of ODM information in symbolic form for each service being reported upon by the ODM. The original representation of ODM data used a monolithic relation for each service. One ODM fact was kept in working memory for every ongoing service, that is working memory contains the latest service snapshot via the latest ODM fact. Situation data requiring longer term memory for a service were and still are kept in other relations and data structures. These auxiliary structures handle situation data such as "Is the signal strength rising, falling or steady from ODM to ODM?".

Because the ODM fact is time-tagged, the ODM fact was guaranteed to change in every data cycle even if the represented service was in steady state. In each data cycle, the ODM fact for each service was retracted and reasserted. The ODM relation was matched many places in the FIESTA code. There was a great deal of wasted computation every data cycle retracting, reasserting and rematching this fact for services which had not changed substantially.

Several modifications were made to the ODM relation. Figure 4 is a before and after picture of the ODM relation for a particular type of Shuttle SSAR (Single Access, S-band, Return) service. *Defrelation* is an ART template for facts. It consists of variables (preceded by question marks) and literals. The left side of Figure 4 defines an ODM good for all service types and leaves room for service-specific details in the *\$?remainder* construct. Below that, the commented part of the relation corresponds to the remainder for the Shuttle SSAR service. It was not part of the ART code but was used as low level documentation. The general form of pairing data tags with values was necessary because ODM was a

⁵Another stamp called a "message id" is present in ODMs and FIMS reports. We bundle it with the time stamp for this discussion.

⁶We use *fact* loosely here. In actuality, the FFEP turns out s-expressions that become facts upon assertion into ART's fact base.

FIGURE 4: OLD AND NEW RELATIONS FOR SHUTTLE SSAR ODM SERVICES

variable length relation. Variable length relations slow the pattern matching process since locations of specific data are unknown in generalized patterns.

Redesigned relations were smaller and more numerous than their predecessors but represented the same HSM data. Relations that were broken up needed to be connected to each other in some way. We used a "key" much like that used for a relational database. Three basic breakups considered were organizing relations as:

1. (key tag item) ; every data item in its own relation
2. (key old-relation-minus-time-stamp) and (key time-stamp)
3. an intermediate form with groupings of data based on match characteristics and relative frequency of change

We had few analytical tools for choosing among the alternatives. We did not wish to allocate the manpower to implement all 3 designs (or design families in the case of alternative 3) for experimentation.

The first alternative was attractive because many items change rarely. Their changes would not affect other relations at all. The second was attractive because it removed the time-stamp, which was guaranteed to change frequently, from everything else. The third alternative was attractive because we knew enough about the data items and their relative frequency of change to group them by items that were likely to change together or be matched together or both. We ruled out alternative 2 because it left us with variable length relations which were known to be inefficient. We ruled out alternative 1 because of a simple experiment that suggested that it was faster to assert and pattern match upon one large fact rather than several small ones (all

items being matched). Informal experimentation and informed guesses were the basis of our choice of alternative 3. The ODM relation on the right side of Figure 4 is one of the results.

The redesigned relation of Figure 4 represents a number of changes:

- All HSM relations were made of fixed length; the *\$?remainder* constructs were eliminated. This made fixed length patterns possible and eliminated excess pattern matching induced by pattern wild cards that matched variable numbers of tokens. Note that making HSM relations of fixed length allows us to drop the data tags in the relations. We have elected not to do so⁷ because 1) the code is somewhat self-documenting with the tags in place; it would be much harder to maintain otherwise and 2) the tags are sometimes used in explanation and justification mechanisms.
- The service IDs of the new HSM relations are the keys that tie together the new relations. Service IDs⁸ effectively condense 5 pieces of information on each service into one symbol:
 - TDRS
 - support identification code or SUPIDEN — a designator for a particular user and category of service, unique for each Shuttle mission and for other spacecraft
 - service type — forward (earth to spacecraft) or return (spacecraft to ground)

⁷Data tags were dropped for the simplest and most frequent relations—the "ID" relations for timestamps and antenna pointing angles.

⁸Service IDs are generated by the FFEP when it sees service scheduling orders (SHOs); they are later inserted in other HSM facts. Matching one short symbol is obviously more efficient than matching a list of several data items.

- service subtype — includes three additional pieces of information: radio frequency band, single/multiple access and TDRS antenna number
 - scheduled time — needed to differentiate different services that use the same physical resources at different times
- Representation of time was changed to one integer that fits in 32 bits: the number of seconds since midnight Dec. 31, 1979 (GMT) instead of the previous 5 tokens required.
 - Some of the data tags have been shortened. When tags were longer and data was not filtered, approximately one quarter of the Lisp machine's processing power was devoted to the process that read s-expressions over the network. This was reduced linearly with the number of characters by reducing tag length at a slight expense in code readability.

Some other things should be noted about the reorganized Shuttle SSAR ODM of Figure 4⁹. The relation has been broken into 6 relations. The first three are generic to all ODM relation sets. They are separated because of the frequency with which they change. The other three are more specific to single access ODMs and to Shuttle SSAR ODMs. The considerations for grouping data items as we have done is:

ODM contains information summarized in the service ID for FIESTA purposes. (Each SUPIDEN maps to a unique vehicle identification code (VIC).) This infor-

mation does not change over the life of a service.

id-ODM contains a time stamp and a message identifier (an integer), both of which change every data cycle. These are natural to group together.

pointing-ODM contains the pointing angles of the TDRS antenna single access dishes or multiple access beams. These data typically change every few data cycles for low orbiters as they pass under the TDRS. The resolution is to one tenth of a degree. These data are matched only a few places in the antenna pointing diagnosis logic; the penalty for changing them—retracting and reasserting—is not too great.

SA-ODM specific to all single access services; contains information which does not change over the life of a service.

Sh-SSAR-config-ODM information that changes infrequently over service life. These data are control settings that change as a result of operator actions.

Sh-SSAR-perf-ODM the performance data of most interest to FIESTA at fault time. Changes from nominal to non-nominal operation and vice versa are reflected in changes in the data of this relation. One possible refinement is to put signal strength in its own relation. It is the only variable in this relation with more than 2 possible values. Changes in the other variables are always significant; signal strength can and does change somewhat in nominal operation without being symptomatic of a fault.

Data Filtering

Data filtering is done in the FFEP. Once relations were broken down as documented above,

⁹A few minor differences in the before and after relations are attributable to domain expansion. Between these two sets of relations, diagnostics for non-Shuttle spacecraft services were added to FIESTA's knowledge base. Some data tags were generalized in that process.

filtering was straightforward. Any data that changes from data cycle to cycle causes its entire relation to be transmitted to the Lisp machine. When a new fact arrives to replace an old one, the expert system first retracts the old one and then asserts the new one. Filtering of redundant data yielded the greatest speedup by eliminating various productions from unnecessarily moving in and out of the conflict set upon fact retraction/reassertion. This was possible because relations that change infrequently were separated from other relations in relation redesign.

Other Modifications

[clay87] is a good overview of optimizing Rete algorithm applications in general and ART applications in particular. It focuses on things such as ordering of patterns and joins. We used many of its guidelines in optimizing various pieces of FIESTA. Chief among its recommendations are elimination of wild cards in patterns. This we have by and large achieved. Miscellaneous modifications made in the name of efficiency include reducing the number and use of global variables (to make garbage easier to recognize and collect) and a revamping of transient displays' graphics. Some graphics were originally implemented using a number of ART rules to call graphics primitives. We rewrote them in Lisp using Symbolics' object-oriented system (Flavors). This tied us closer to the Lisp machine but bought us some speed. Every ART rule requires some overhead; rule-driven graphics also cluttered our pattern/join nets unnecessarily. Moving procedural operations to Lisp code on the Lisp machine is almost guaranteed to be more efficient. Revamping of graphics was also used as an opportunity to provide some abstraction and general cleanup of the transient display system.

Also during the period in which the benchmarks of Section 6 were conducted, several

efficiency-oriented modifications were made to ART. Inference Corp. reduced the amount of garbage ART creates, made the menu system more efficient and provided a rule compiler. (Pre-compiled rules eliminate the garbage of rules compiled "on the fly," giving us a smaller image which translates to fewer paging and garbage collection operations.) Pattern and join net operations were also made more efficient (release 3.0 of ART). Symbolics upgraded its microcode and system software in the same period, making some operations more efficient. Rigorous benchmarking would eliminate the variable software environment; we did not care as long as we knew that the modifications we made were moving us in the right direction.

Possible Improvements

One necessary modification to be made to go on-line is provision of an HSM buffering mechanism for those times when FIESTA lags real time. In the off-line testbed, we have the luxury of temporarily halting the reading of input data. Some buffering is provided in the current system by the mechanisms that move data from the FFEP to the expert system; these are not adequate for the on-line system.

Possible improvements we could make include:

- further refinement and more rigid benchmarking of relation designs
- experimenting with the Lisp machine process scheduler to devote more time to the tasks that need it
- coding more algorithms in Lisp instead of ART rules. The operator login procedure is a good example. It is basically procedural code implemented in ART rules that add clutter and size to the pattern/join net and to the fact database.

- handling temporary windows as *resources*; resources are objects that are explicitly allocated/deallocated by the application instead of being created in general-purpose memory and collected by the garbage collector.
- eliminating the ART studio altogether. It is written to be portable; we can buy efficiency by further committing ourselves to a given window system.
- reexamining our software/hardware environment in general

5 Implementation Plan

FIESTA was developed in a series of "builds." Successive builds were characterized by increasing levels of complexity and attention to the demands of the operational environment. The first 3 builds concentrated on concept validation (Build I), requirements definition (II) and operational issues (III). Early builds resolved these issues. Build IV focused on real-time performance issues. Cleanup and restructuring was also made possible. This is not to say that performance issues were ignored altogether in early builds.

Modifications identified in Section 4 were also phased (Figure 5). Phase contents are outlined here to make the results of Section 6 understandable:

Baseline Build III system was ported to the latest vendor software (including ART 3.0).

Build IV A Data tags were shortened, decreased LAN traffic and memory requirements.

Build IV B HSM relations were redesigned.

Build IV BBIN Rule files were compiled.

Build IV C Redundant HSM data was filtered by the FFEP.

Build IV D The menu system was rebuilt; data tags were removed from the "ID" relations.

Some more general cleanup, e.g., reordering of patterns and joins, has occurred throughout FIESTA development, especially Build IV.

Gradual modification made changes more manageable and allowed measurement of the effects of individual modifications. The ordering of changes reflected dependencies and convenience, not expected benefits. For instance, shortening of parameter names occurred before HSM relation redesign to limit the number of patterns changed. Whenever possible, developments were made in parallel. For example, redundant HSM filtering was built in conjunction with FFEP HSM relation redesign.

6 Testing and Results

Three basic tests were run over most of the builds. These tests sent HSM data from the FFEP as fast as the expert system could handle them. Test 1 was designed to determine a baseline for nominal processing, or, in other words, to see how long the absolute minimum processing would take. It therefore monitors a stream of HSMs known to show nominal operation. No monitoring displays and no operator interaction of any type were used.

Test 2 and 3 looked at relevant time periods within an event to see how long specific activities would take. Both tests cover the same data, however Test 3 brought up a monitoring display and had the operator interact with the system, whereas Test 2 did not. Therefore Test 2 gave a baseline for selected activities, and Test 3 gave an indication of the cost of optional processing. The results presented

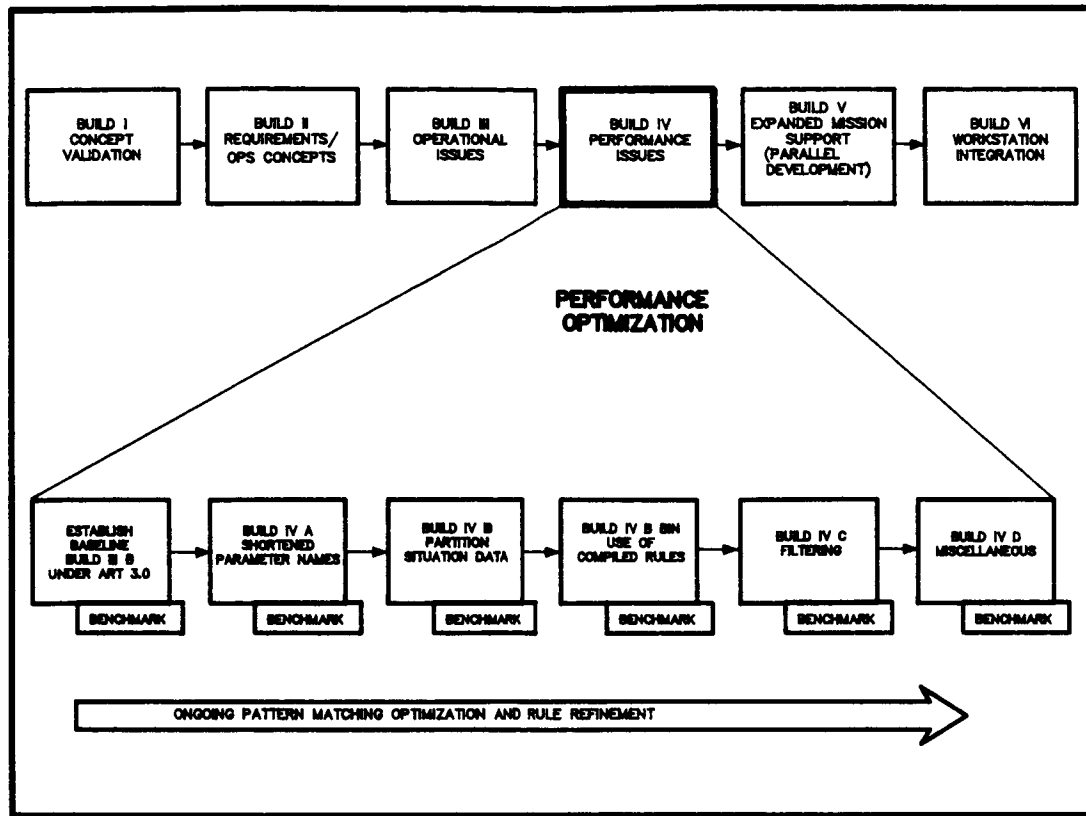


FIGURE 5: PHASED IMPLEMENTATION

ACTIVITY	SIMULATION TIME	ART 2.0	ART 3.0	BUILD 4A	BUILD 4B	BUILD 4B BIN	BUILD 4C	BUILD 4D
TEST 1: NO DISPLAYS, NO INTERACTION								
NOMINAL SERVICE	8:25	--	--	14:53	10:15	9:38	2:41	2:19
TEST 2: NO DISPLAYS, NO INTERACTION								
SERVICE STARTUP	:45	2:27	3:16	2:33	2:04	1:33	1:03	:56
ACQUIRE SERVICE	2:41	4:11	6:12	5:21	4:16	3:26	1:19	1:08
DIAGNOSTIC EPISODE	1:55	5:35	5:19	5:01	3:56	3:06	1:24	1:22
TEST 3: 1 COCKPIT DISPLAY, MINIMAL INTERACTION								
SERVICE STARTUP	:45	2:50	3:20	2:51	2:20	1:41	1:11	1:07
ACQUIRE SERVICE	2:41	4:29	6:23	5:45	4:34	3:37	1:28	1:12
DIAGNOSTIC EPISODE	1:55	6:30	5:58	5:34	4:35	3:22	1:41	1:33

TABLE 1: FIESTA BENCHMARKS

4/28/1988/SM7000

MIS88\SM7000

(Table 1) should be compared to the "simulation time" in the first column—the wall clock time of the original HSMs.

Longer term tests were run at the conclusion of Build IV D. Test 4 ran through a Shuttle event lasting 1:00:59 from start to finish. The system was run as it would be in the operational environment. HSMs were sent to the expert system in real time (as opposed to the other tests which sent them as fast as possible). Six monitoring displays were up throughout the event. The event contained 7 diagnostic episodes and two service handovers. All notifications were acknowledged by the operator. FIESTA was able to process the event in 1:01:10. The extra 11 seconds were used after the last message was transmitted in order to clean out the database when the event was over. The system fell behind its HSM input stream by no more than 30 seconds whenever fault diagnosis was initiated, but the system always caught up with its inputs once a fault cause and location were found.

A fifth test was designed to run through seven consecutive hour long events in an "operational" mode (real-time HSM stream, monitoring displays up, notifications acknowledged). Unfortunately the FFEP crashed during the sixth event. After approximately five hours (5:38) of continual processing, FIESTA was still keeping up with the incoming data, except during fault diagnosis, but would still "catch up" with the data flow afterwards. All tests were run with ephemeral garbage collection (GC) on and dynamic GC off¹⁰. Details of these results are available in [sti87b].

¹⁰Ephemeral GC collects short lived objects; dynamic GC collects longer lived objects. Dynamic GC will probably be used in the on-line system

7 Conclusions

Despite the fact that it is not a hard real-time system, the current FIESTA more than meets its performance requirements as best we can test it in the off-line environment. We do not see the justification for radical modifications in the name of efficiency now. Faster and less expensive hardware is being made available in the marketplace; other FIESTA cleanup is being done in preparation for the on-line system. We are more concerned with getting FIESTA on-line and verifying/validating it in an on-line testbed.

We have concluded that manually optimized systems are in general fragile. The optimization process we used was entirely too labor-intensive. Better technology must be available for optimizing knowledge-based systems and guaranteeing their performance.

Acronyms

ART Automated Reasoning Tool¹¹

FFEP FIESTA Front End Processor

FIESTA Fault Isolation Expert System for TDRSS Applications

FIMS Fault Isolation Monitoring System

GC garbage collection

HSM High Speed Message

KBS Knowledge-Based System

NASCOM NASA Communications Network

NCC Network Control Center

NGT NASA Ground Terminal

ODM Operations Data Message

OPM Operations Message

¹¹ART is a trademark of Inference Corp.

SHO Scheduling Order
SN Space Network
SSAR S band, Single Access, Return
SUPIDEN Support Identification Code
TDRS Tracking and Data Relay Satellite
TDRSS TDRS System
VIC Vehicle Identification Code
WSGT White Sands Ground Terminal

Acknowledgements

The overall support and encouragement of Paul Ondrus, Anthony Maione and Dawn Lowe of NASA/GSFC are particularly appreciated and acknowledged. Nathan Dreon of Stanford Telecommunications played a significant role in FIESTA performance improvement, making necessary FFEP modifications. Mark Grover merits special thanks for stimulating several ideas presented here.

References

- clay87** Clayton, B.D., "ART Programming Tutorial, Volume Three: Advanced Topics in ART," Ch. 7, "Efficiency," Inference Corp. 1987.
- forg80** Forgy, C.I., "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," *Artificial Intelligence*, 19, 1980.
- laff88** Laffy, T.J., Cox, P.A., Schmidt, J.L., Kao, S.M., Read, J.Y., "Real-Time Knowledge-Based Systems," *AI Magazine*, vol. 9, no. 1, Spring 1988.
- less88** Lesser, V.R., Pavlin, P. and Durfee, E., "Approximate Processing in Real-Time Problem Solving," *AI Magazine*, vol. 9, no. 1, Spring 1988.
- lowe87** Lowe, D., Quillin, B., Matteson, N., Wilkinson, B., and Miksell, S., "FIESTA: An Operational Decision Aid for Space Network Fault Isolation," 1987 Goddard Conference on Space Applications of Artificial Intelligence and Robotics, May 1987.
- miks87** Miksell, S., Quillin, B., Lowe, D., "Network Fault Diagnosis: Knowledge Representation Using Parallel Reasoning," *Proceedings of the 1987 Expert Systems in Government Conference*, IEEE Computer Society, Oct., 1987.
- norm85** Norman, D.O., "Reasoning in Real-Time for the Pilot's Associate: An examination of a Model Based Approach to Reasoning in Real-Time for Artificial Intelligence," *Masters Thesis*, Air Force Institute of Technology, Dec., 1985.
- orei85** O'Reilly, C.A. and Cromarty, A.S., "Fast is not 'Real-Time': Designing Effective Real-Time AI Systems," *Proceedings of SPIE*, 1985, J.F. Gilmore, editor.
- sti87a** Stanford Telecommunications, Inc., *Functional Requirements for the FIESTA On-line Testbed*, TR870114, Rev. A, 20 Oct., 1987.
- sti87b** Stanford Telecommunications, Inc., *Technical Report for the FIESTA Performance Optimization Subtask*, TR870133, 15 Oct. 1987, Rev. A.

Mission Telemetry System Monitor:
A Real-Time Knowledge-Based System

Samih A. Mouneimne

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109

ABSTRACT

The Galileo Mission Telemetry System (MTS) has a cluster of computer subsystems configured as a star network. The MTS handles the real-time processing of spacecraft telemetry and ground monitor data. Large volumes of status and fault messages are generated as a result of changes in the system environment. These messages are triggered by the conditions that exist on any one particular subsystem or device. The order of message generation is in time sequence and does not always correlate to the function sequence of active processes. A significant number of messages provide context with varying degrees of uncertainty. As such, highly skilled telemetry controllers are required to regularly go through large volumes of messages generated by the MTS to identify, diagnose, and isolate faults.

A knowledge-based system prototype is being developed to monitor the Galileo Mission Telemetry System performance. The system design approach features temporal reasoning, uncertainty management, and intelligent graphic user interfaces.

PRECEDING PAGE BLANK NOT FILMED

INTRODUCTION

A study conducted in 1986-87 of NASA/JPL Mission Operations to identify tasks that would benefit from the application of artificial intelligence as an emerging technology concluded that:

- a. The present approach to conducting mission operations successfully requires a large staff of engineers, controllers, operators, and technicians for each mission.
- b. The present schedule calls for launch and operation of several missions within a relatively short period of time.
- c. The projected requirements for mission operations in the next several years, if performed as currently planned, will result in the need to double the mission operations staff.
- d. The economic constraints of these scheduled missions have created the need for new approaches and new tools to enhance the reliability and productivity of JPL mission operations.
- e. Certain tasks within mission operations would benefit significantly from the application of knowledge-based systems technology. The Galileo Mission Telemetry System Monitor, the subject of this paper, was one of them.

THE TARGET SYSTEM

The JPL Galileo Mission Telemetry System (MTS) is comprised of five computer subsystems in a star network configuration (see Figure 1). Each subsystem has a set of peripheral hardware for data entry, storage, manipulation, and display. Each computer handles specific processing functions relating to spacecraft downlink telemetry data such as input, frame synchronization, data extraction, engineering and science decommutation, display of science status channels, and science experiment processing. One of the processors, the SIO (see Figure 1), acquires streams of literal status and fault conditions on each of the computer subsystems, the Star Switch Controller, the network configuration, and the software processes currently executed. The order of message generation at the SIO is in time sequence and does not necessarily correlate to function sequence performed, or relate to interaction among anomalies. Error conditions are triggered within each of the network subsystems by the conditions that exist on any one particular subsystem or device. Although existing expert knowledge is available, the association of multiple faults as a failure vector and fault trend analysis are not performed under the present conditions. The volume of information generated that requires interpretation for diagnostic purposes is relatively large for effective real-time monitoring. A short time interval is sufficient to drive controllers' tracking capabilities to overload conditions. One of the objectives of the knowledge-based system (KBS) is to overcome this difficulty.

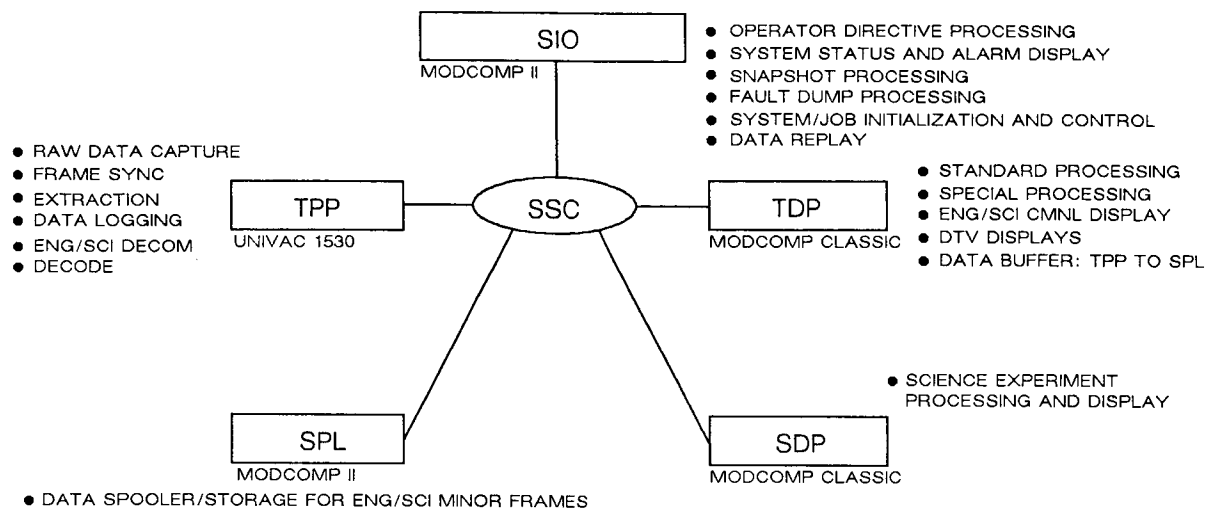


FIGURE 1. GALILEO MISSION TELEMETRY SYSTEM REPRESENTATION

THE HOST ENVIRONMENT

The knowledge-based system prototype is being developed on a Sun3/160 workstation utilizing ART 3.1 shell and Common Lisp. The plan is to port the prototype on a Sun4/260C workstation for demonstration purposes. The choice of the hardware platform was in concert with the present approach of utilizing network workstations for JPL's future Space Flight Operations Center (SFOC).

USER INTERFACE

The knowledge-based system prototype utilizes graphics extensively to provide controllers with a continuous visual representation of the Mission Telemetry System performance status. Graphic icons are represented either as frames or as images. Frames are utilized when icons are involved in reasoning while images are for static representation. This approach is guided by efficiency considerations. Changes of graphic icons and windows are triggered by MTS input messages to the knowledge-based system via the SIO, mouse clicking, or keyboard entry. For the initial phase of the prototype development, actual MTS messages built into the data base will be utilized. The MTS system is represented by two types of hierarchical graphic display configurations: data flow and network links. The data flow representation emphasizes the telemetry data stream and the active software processes. The network links represent the physical hardware such as computers, peripherals, and communication lines. The target system graphic representations, as user interfaces, are at the system, sub-system, and peripheral levels. Distinctive graphic icons pointing to the probable location of faults provide simple and efficient cues to the telemetry controllers. All explanations are provided through textual window(s).

One of the system user interface features is a user-system dialogue initiated either by the user or the system. On the one hand, the user may start a dialogue, such as calling for an explanation of a decision

made by the system. On the other hand, the system may initiate a request to the user, such as the confirmation of a device status, to clean irrelevant data from the data base. The dialogue between the user and the system is on a non-interference basis. The KBS will not halt its operations pending user responses but will continue processing of input messages and periodically attend to user dialogue as a lower priority service.

KNOWLEDGE REPRESENTATION

Knowledge in the prototype relating to the MTS fault diagnosis is represented in three categories:

1. Knowledge about the MTS network configuration.
2. Knowledge about MTS message taxonomy.
3. Knowledge about MTS software telemetry processes.

The structure of the MTS network is represented into a schema language that views the system as a hierarchy of subsystems, interfaces, and peripherals that share, through inheritance, common attributes and are suitable for reasoning. The set of message classifications and subsequent search for status, faulty components, or software process errors can be identified consistently in a streamlined knowledge-base. The knowledge-based system tracks the MTS network configurations that map into specific active telemetry software processes. For the initial phase of prototype development, actual MTS messages built into the data base will be utilized. The input method is through timed file read function or keyboard entry. The timing is required to simulate the actual MTS real-time environment.

TEMPORAL REASONING

The existing Mission Telemetry System messages are asynchronous events providing significant representation of the system's state(s) over varying time intervals. In addition, heuristic knowledge asserts that message context may change according to temporal generation. To a large degree this is a manifestation of the change in the underlying system state or any of its objects. The principle of persistence is utilized to reference the continuation of the state of any object unless changed by transition rules. Transition rules control the change of message context and object states. For the purpose of this KBS, reasoning over time is treated according to the following approach:

- (A) Messages that recur over short time intervals ($\sim \leq 10$ s) and originate from the Starswitch Controller (Figure 1), one of the MTS computer subsystems, or multiple computer subsystems. This type has two probable outcomes: software error or hardware fault. This message type points to significant probability of failure (including data queueing) if originated from a specific computer subsystem. The KBS tracks the temporal state and the association among fault messages that maps into the current configuration of the network subsystems. For messages that originate from the intercomputer network task, the KBS looks for confirmation from one or more linked subsystem on the network. Lack of confirmation would trigger the application of uncertainty rules.

- (B) Messages that recur over long time intervals (~ 11–300 s) and originate from the same source as type (A). The KBS creates time-tagged message files in the data base. These files are utilized for trend analysis to predict probable hardware failure or software anomalies.
- (C) Messages that are characterized by a single occurrence over a specific time interval (~ >300 s) and originate from either a status or possible fault condition in the MTS. Status messages point to the network configuration and the current software processes that are executed on any one of the MTS subsystems. Based on heuristic expert knowledge, fault messages that do not recur within certain time intervals are considered an anomaly of the existing system. As such, all information in the data base relating to this message is declared irrelevant to the immediate system objectives and hence retracted from the data base or stored in archival files.

UNCERTAINTY

The MTS has a number of fault message types whose context has inherent levels of uncertainty. This is particularly true for fault messages that are construed to have multiple sources of failure. In a considerable number of cases, faults and, subsequently, fault messages are induced by one or more conditions on the system network. As such, the use of conditional probability construct is not appropriate where MTS events are considered dependent. The KBS prototype uncertainty management approach is to limit the assignment of certainty factors to various possible outcomes of relevant messages. This is consistent with the heuristic knowledge available. However, the uncertainty rules are not invoked until temporal reasoning is performed.

FUTURE WORK

Two additional efforts are planned for this knowledge-based system: 1) evaluate, modify, and build the prototype for real-time operations, 2) include the modified system as an agent of a distributed knowledge-based system research effort.

CONCLUSIONS

The prototype effort is a test ground of the use of knowledge based systems in support of JPL mission operations. While the target system provides rich environment for the application of knowledge-based systems, the use of existing system messages as a diagnostic source of information limits the system capability to the extent of their scope and representations. However, the chief purpose of this and similar efforts is to demonstrate an emerging technology potential for the next generation of mission operations systems.

REFERENCES

1. Doyle, J., "A Truth Maintenance System", Artificial Intelligence, 1979, North-Holland.
2. "Telemetry Subsystem: Software General Design Document", JPL 625-650-221011, 1982.
3. Allen, J. F., "Maintaining Knowledge about Temporal Intervals", *Communications of the ACM*, 1983, 832-843.
4. Hayes-Roth, F., Waterman, D. A., and Lenat, A. B., "Building Expert Systems", 1983, Addison-Wesley Publishing Company, Reading, Massachusetts.
5. Sowa, J. F., "Conceptual Structures", July 1983, Addison-Wesley Publishing Company, Reading, Massachusetts.
6. Cohen, P., "Heuristic Reasoning About Uncertainty: An Artificial Intelligence Approach", 1985, Morgan Kaufman, Inc., Los Altos, California.
7. Siemens, R.W., Golden, M., and Ferguson, J. C., "Starplan II: Evolution of an Expert System", p. 844, Vol. 2 *Proceedings of AAAI-86 National Conference on Artificial Intelligence*, August 11-15, 1986.
8. Mouneimne, S. A. and Carnakis, J. M., "Applying Artificial Intelligence to Ground Mission Operations", JPL D-4784, August 1987.
9. Shoham, Y., "Reasoning About Change", 1988, MIT Press.

Image Processing and Machine Vision

**Low Level Image Processing Techniques Using The Pipeline
Image Processing Engine In The Flight Telerobotic Servicer**

**Autonomous Image Data Reduction By Analysis And
Interpretation**

**An Automated Computerized Vision Technique For
Determination Of Three-Dimensional Object Geometry**

**An Interactive Testbed For Development Of Expert Tools For
Pattern Recognition**

**Parallel And Distributed Computation For Fault-Tolerant
Object Recognition**

Range Data Description Based on Multiple Characteristics

Low Level Image Processing Techniques
Using the Pipeline Image Processing Engine
in the Flight Telerobotic Servicer

Marilyn Nashman and Karen J. Chaconas

National Bureau of Standards, Gaithersburg, MD 20899

Abstract

This document describes the sensory processing system for the NASA/NBS Standard Reference Model (NASREM) for telerobotic control. This control system architecture has been adopted by NASA for the Flight Telerobotic Servicer. The control system is hierarchically designed and consists of three parallel systems: Task Decomposition, World Modeling and Sensory Processing. The paper will concentrate on the Sensory Processing System, and in particular will describe the image processing hardware and software used to extract features at low levels of sensory processing for tasks representative of those envisioned for the Space Station such as assembly and maintenance.

1. Introduction

The NASA/NBS Standard Reference Model (NASREM) architecture for the control system of the Flight Telerobotic Servicer defines an architecture for telerobotics based on concepts developed in other research programs. It incorporates artificial intelligence theories such as goal decomposition, hierarchical planning, model driven image analysis, blackboard systems and expert systems [1]. The multiple processes of the system are hierarchically structured. Each process is considered to be arranged vertically in a hierarchy which decomposes complex tasks into progressively simpler objectives. In addition to the vertical structure, the system is also partitioned horizontally into three sections: Task Decomposition, World Modeling, and Sensory Processing (Figure 1).

The Task Decomposition System is responsible for monitoring tasks, planning, and control servoing of the robot's manipulators, grippers, and sensors. The complexity of each function is determined by its position in the hierarchy [4, 11]. The World Model is responsible for maintaining the best estimate of the current state of the system and of the world at any given point in time. It is responsible for maintaining models of objects and structures, maps of areas and volumes, lists of objects describing features and attributes, and tables of state

variables describing the system and the environment. The Sensory Processing System is responsible for gathering sensory information from multiple instances of various sensors [8], enhancing that information [9], recognizing features, objects, and relationships between objects, and determining the correlation between observations and expectations.

Section 2 of this paper details the lower layers of the Sensory Processing System hierarchy. In Section 3, a parallel hardware system that is particularly well-suited for performing low level processing tasks is described. Section 4 explains a number of techniques employing local operations that are used to enhance data and extract features and that have been implemented on parallel hardware.

2. Sensory Processing in the NASREM Architecture

The Sensory Processing System (SPS) in the NASREM architecture [1] is designed so that data flows bidirectionally between the levels of the Sensory System and bidirectionally between the Sensory System and the World Model (Figure 1). The SPS is designed to operate in both a bottom-up (data driven) and a top-down (model driven) mode. The World Model contains both *a priori* information and updated information required to perform sensory processing tasks. At each level of the Sensory Processing hierarchy, information will be sent to the World Model. This information will be made available to the Task Decomposition module at the level in which it is needed.

The system is divided into four levels: Data Acquisition, Low Level Processing, Intermediate Level Processing and High Level Processing. This organization parallels that described in [2]. The Data Acquisition Level serves as an interface between the environment and the Sensory Processing System. It gathers raw information (readings) from each of the sensors. Depending on the complexity of the data, this information may be stored directly into the Servo Level of the World Model or used for further processing at the next level [8]. The Low Level Processor performs point-by-point operations to enhance the raw data and to perform local feature extraction. Its output is passed to the World Model at the Prim Level and/or to the Intermediate Level Processor. The Intermediate Level Processor is responsible for providing symbolic descriptions of regions, lines and surfaces that have been extracted from Low Level Processing. This data is passed both to the World Model and to the next SPS level. Lastly, the High Level Processor is responsible for interpreting and labeling the "intermediate symbolic representation" [2] and for updating the contents of the World Model with the most current knowledge about the position and orientation of objects.

3. The Parallel Image Processing Engine

The information processed by the Low Level Processor is in the form of arrays of data received from cameras, ranging sensors, or tactile array processors [9]. A typical image can consist of between 16K (128 x 128) bytes and 1M (1024 x 1024) bytes of information. Because of the large amount of data to be processed and the need to process that data as quickly as possible, most serial computers cannot meet the requirements of low level processing. Parallel computers have been developed in recent years to specifically fulfill the need of real-time processing of image data [6, 7], and although the machines differ in architectural design and implementation, they share the capability of being able to process an

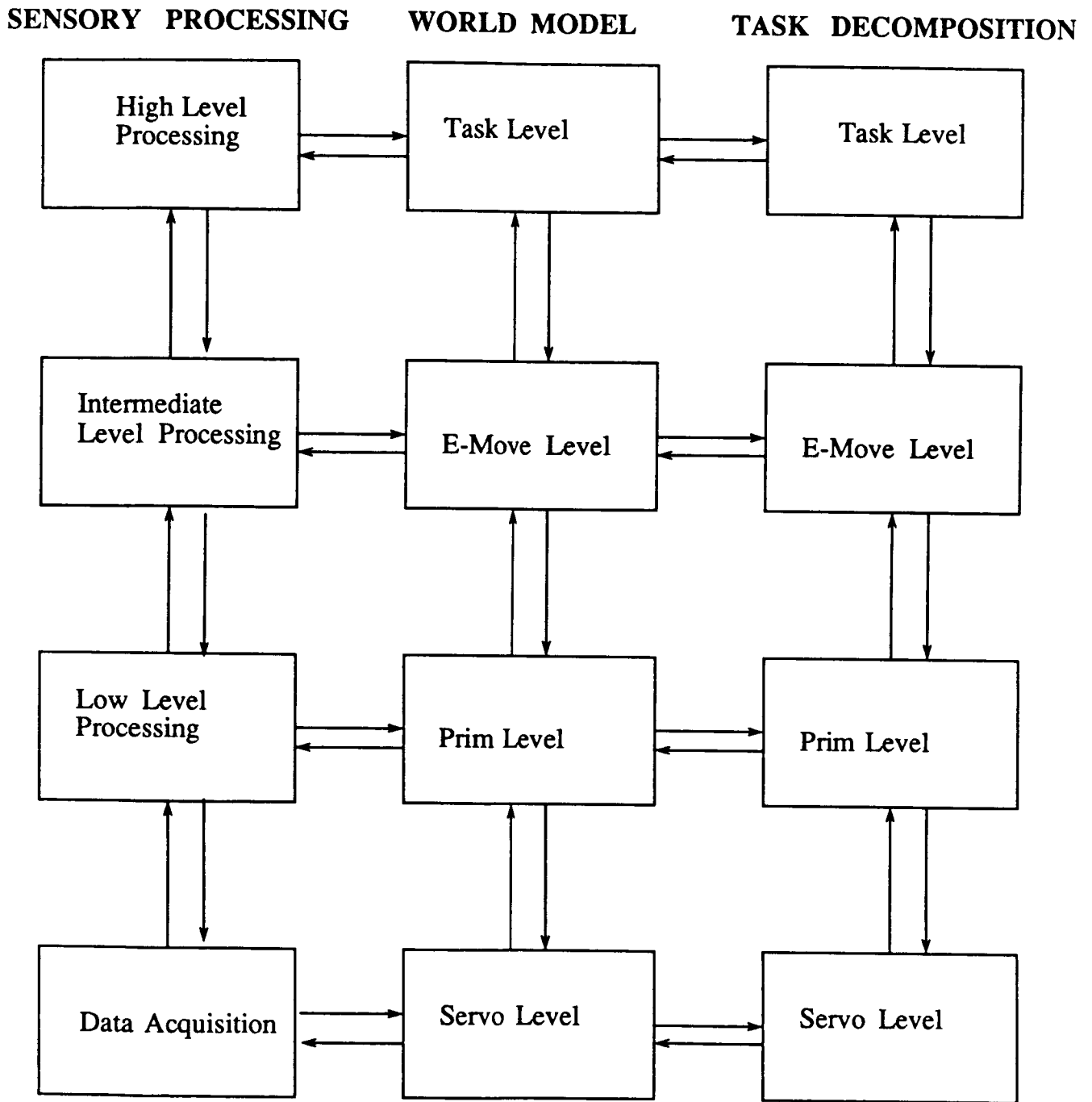


Figure 1. NASREM Hierarchy

entire image or region of an image in real-time. Parallel processing is especially applicable to low level image processing. The data structure used at this level is the image itself, a spatially indexed image of points which correspond to gray scale intensity values. All parts of the image are treated in the same way, and in general, no effort is made to distinguish between different parts of it. Local operations depend only on corresponding elements between images or on combinations of adjacent elements of an image (Figure 2). Computations tend to be simple arithmetic, algebraic, or logical operations, and typically a low number of computations per pixel is required [5]. Parallel processors are also suited to multi-resolution representations and processing techniques.

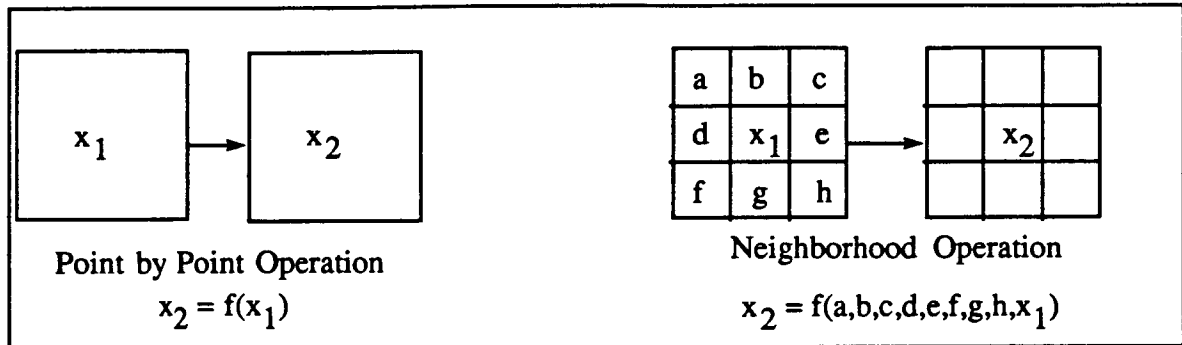


Figure 2. Local Operations

Many local data enhancement techniques can be implemented on the Pipelined Image Processing Engine (PIPE) developed at the National Bureau of Standards and manufactured by Aspex, Inc. Some features of PIPE are discussed here, but the reader is referred to [6, 7] for a more detailed description of the system. PIPE acquires its images in real-time from analog sources such as cameras, video tapes, and ranging devices, as well as digital data sources. Its output can be directed to video monitors, symbolic mapping devices, and higher level processing systems. All inputs and outputs are synchronous with the video rate of sixty fields per second.

The PIPE system is composed of up to eight identical modular processing stages, each of which contains two image buffers, look-up tables, three arithmetic logic units, and two neighborhood operators (Figure 3).

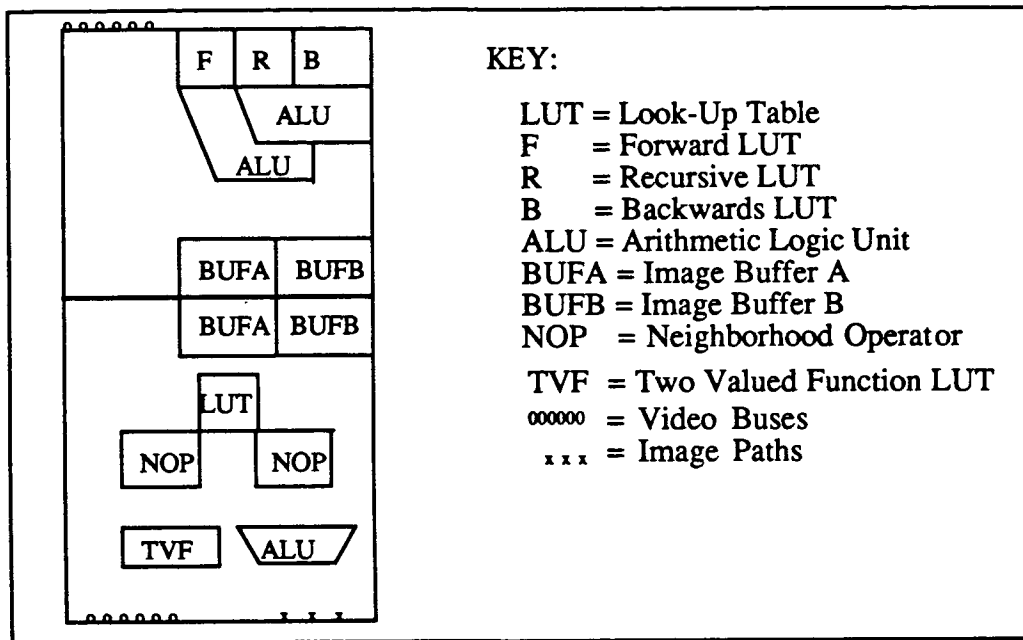


Figure 3. PIPE Modular Processing Stage

A forward path from one stage to the next allows pipelined and sequential processing. A recursive path from a stage output back to its input allows feedback and relaxation processing. A backward path from one stage to the previous stage allows for temporal operations (Figure 4). The images in the three paths can be combined in arbitrary ways on each cycle of a PIPE program, and the chosen configuration can change on different cycles.

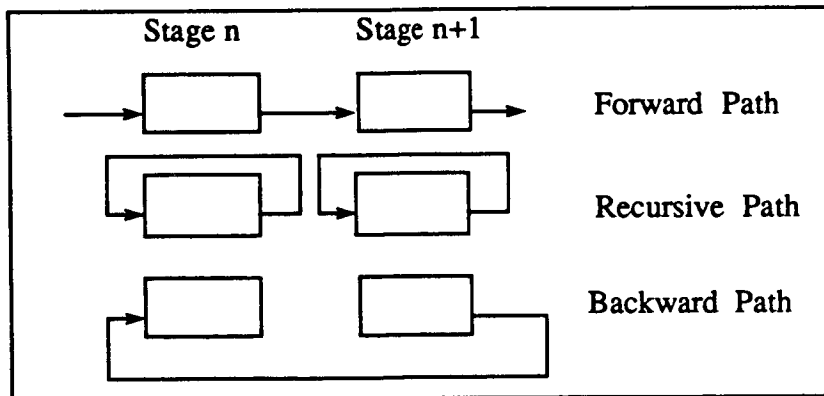


Figure 4. Data Flow Path Between PIPE Stages

In addition, six video buses allow images to be sent from any stage to any one or more stages.

Images can be processed in any combination of four ways on PIPE: point processing, spatial neighborhood processing, sequence processing or Boolean processing (Figure 5). Different processing can occur at individual pixels in the image by using a region-of-interest opera-

tor. All methods can be considered local operations.

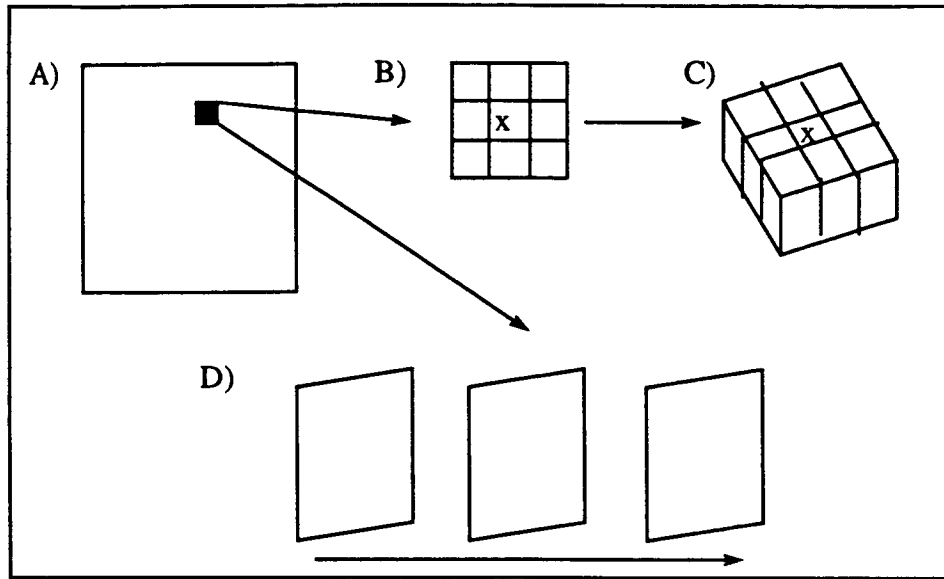


Figure 5. Processing on PIPE: (A) Point (B) Spatial (C) Boolean (D) Sequential

Point processing can be a function of either one or two input images and includes simple arithmetic and logical operations such as scaling, thresholding, converting number systems, etc. Look-up tables resident on each PIPE stage allow the user to perform more complex arithmetic operations, trigonometric operations, comparisons, rotations, etc.

PIPE can perform up to two 3×3 neighborhood convolutions on each stage in parallel. Both neighborhood operators operate on the same image input, but can perform different neighborhood operations. Larger neighborhood convolutions can be achieved by decomposing an odd-sized neighborhood mask into a sequence of 3×3 convolutions. The neighborhood operators can be either arithmetic or Boolean and are performed identically on all locations in the image unless a region-of-interest is specified. Special features are provided to prevent inaccurate computations on the image borders.

Multi-resolution pyramids can be constructed by selecting the "squeeze" or "expand" options as an image is stored or written from a buffer. In the former case, each 2×2 neighborhood of the input image is sampled and written to the output image resulting in an image half the resolution of the original. This process can be repeated to generate successively smaller resolution images. Expanding an image involves the opposite operation by pixel replication and generates successively larger resolution images.

Sequential processing works on a set of multiple images, e.g. sequences of images over time, a stereo pair of right and left images, or multi-resolution images. By taking advantage of the inter-stage paths, images can be combined, compared, sampled or differenced to extract the desired application dependent information.

When performing Boolean processing, each pixel of information is considered to be composed of eight independent bit planes which are operated upon simultaneously. The neighborhood operators can be applied in a Boolean mode, where the output is the combination of

the 3 x 3 neighborhood using local operations on each of the eight bit planes.

PIPE programs are written on a host computer using a software package which is an iconic representation of the hardware to generate microcode. The microcode instructions are downloaded to PIPE, where they are resident during program execution. A software development tool, ASPIPE, allows the user to code the spatial and temporal flow of the data through the hardware and to allocate the look-up tables and PIPE resources to be used. Programs can be edited, saved, compiled, executed, and debugged in this environment. In addition, ASPIPE generates a sequencer file that specifies which micro-operation is executing at each time-cycle. This sequencer also controls branching and looping among microcode instructions during execution.

A hardware interface between PIPE and a high level processor (HLP) has been developed and software has been written to support this interface. In this manner, the results of low level vision tasks are transferred to a serial computer which can perform high level vision tasks of image analysis, recognition, and general decision making which require global information. Since the interface is bidirectional, the HLP can download images or look-up tables directly to any buffer or table on any selected piece of PIPE hardware. In addition, the HLP can select PIPE algorithms by manipulating the PIPE sequencer.

4. Low Level Image Processing Algorithms

Figure 6 is a picture of a truss node suggested for use in assembly of the NASA Space Station. The sockets are attached to the node in various configurations, but the world model has knowledge of the geometry of each instance of any assembly. The appearance of the truss node presents a difficult problem for computer vision: the part is machined of a smooth, highly reflective metal, and the curvature of the node increases the difficulty of obtaining satisfactory information with standard image processing techniques.

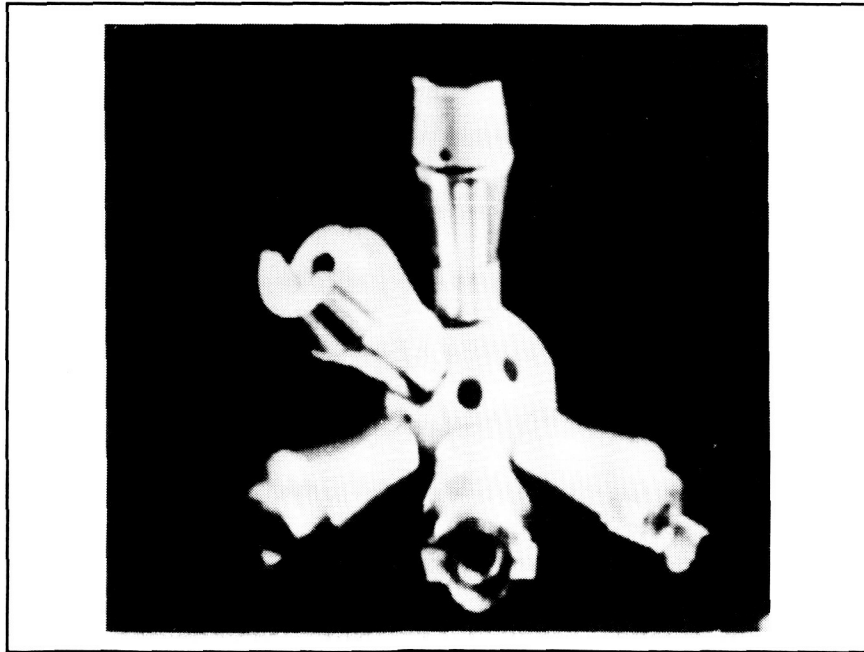


Figure 6. Truss Node Assembly

Binary thresholding of the image fails because of the specularity of the node. Connected component algorithms which segment an image into distinct objects and compute statistical information relative to each object fail because the node is improperly segmented due to highlight and shadow effects. Edge extraction routines provide extraneous information because highlights are falsely interpreted as edges. Figure 7 illustrates the "edges" found in the truss node assembly using a non-maxima suppression algorithm. .

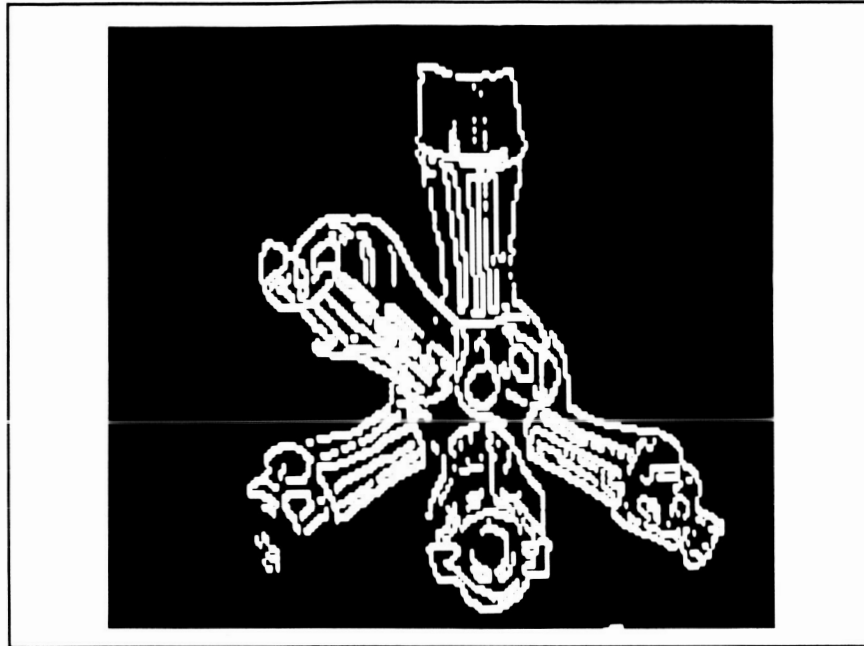


Figure 7. Truss Node Edge Image

To overcome these obstacles, an algorithm was developed on PIPE which makes use of standard edge extraction techniques, image smoothing, and multi-resolution processing. The goal of this algorithm is to provide a connected edge image of the truss node assembly which can be used as input to a connected component algorithm.

The first operation applied in this algorithm involves extracting edges in the full resolution image. A Sobel operator [10] is applied to the image using PIPE's neighborhood operator to extract the x and y gradients at each pixel in the image (Figure 8).

X Gradient Operator			Y Gradient Operator		
-1	0	1	1	2	1
-2	0	2	0	0	0
-1	0	1	-1	-2	-1

Figure 8. Sobel Operator

The magnitude and direction of each edge point are then computed using two-valued function look-up tables. By thresholding the direction image with the magnitude image to remove weak edges, a three pixel wide, binary edge is obtained. In order to thin the edge image, a non-maxima suppression algorithm is applied. This operation involves quantizing the directions of all edge points into one of eight values (Figure 9). The output of this quantization

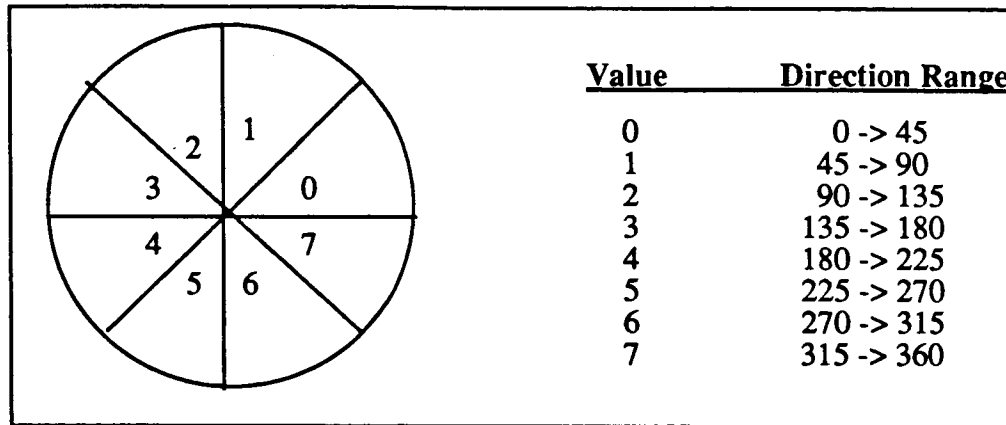


Figure 9. Quantization of Direction Image

is stored in a buffer which is used to determine in which direction to thin the corresponding pixel in the edge image. In this manner, different 3 x 3 masks can be applied to the image depending on the direction of the edge, and all edge points that are not maximum in the gradient direction are eliminated (Figure 7).

In order to remove the extraneous information in the thinned edge image, multi-resolution processing is used. The image is first smoothed using a Gaussian operator [10], and then it is sampled such that each 2 x 2 neighborhood of the original image is averaged to produce one pixel at the next higher level of resolution (Figure 10). The reverse operation is then applied to the smoothed sampled image; it is expanded back to a 256 x 256 image using pixel replication.

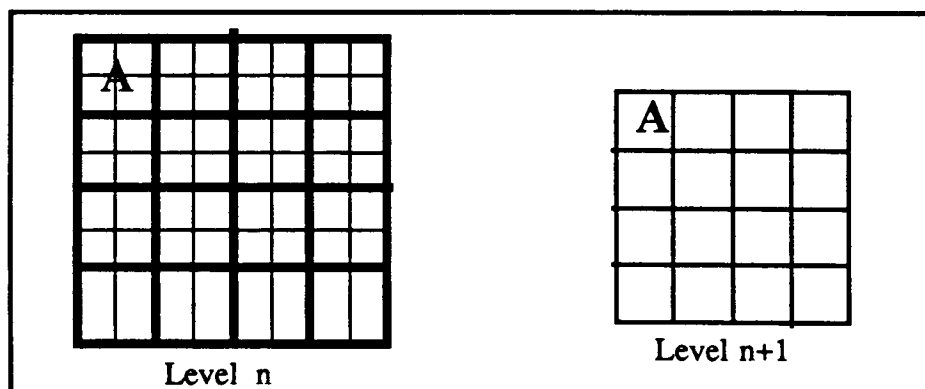


Figure 10. Forming Levels of a Multi-Resolution Pyramid

The result of these operations is shown in Figure 11. The false edges caused by the specu-

larity have been removed and all portions of the truss assembly are connected. Reapplying the Sobel operator to Figure 11 results in a connected edge image (Figure 12), and applying a shrinking algorithm results in a connected, thinned edge image (Figure 13).

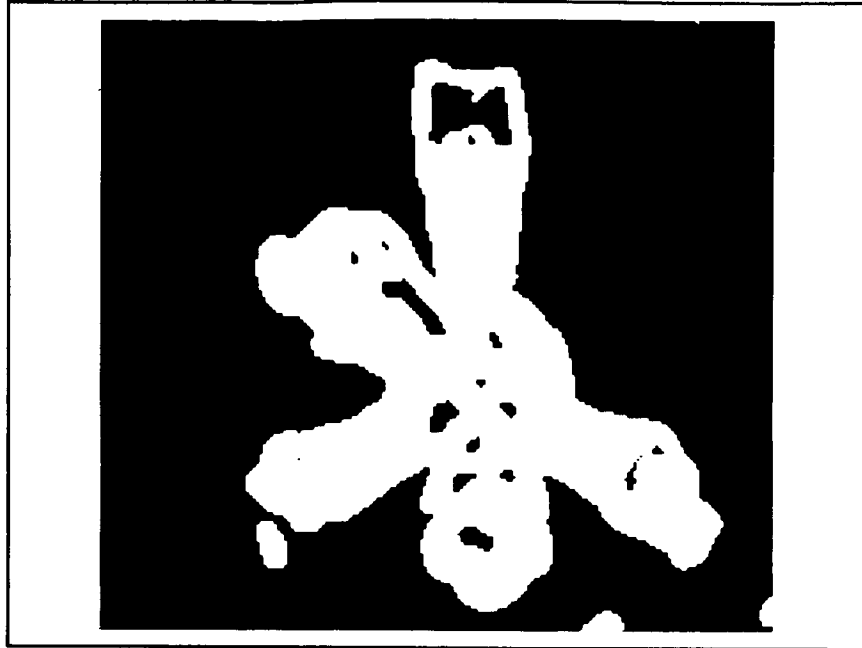


Figure 11. Result of Multi-Resolution Processing

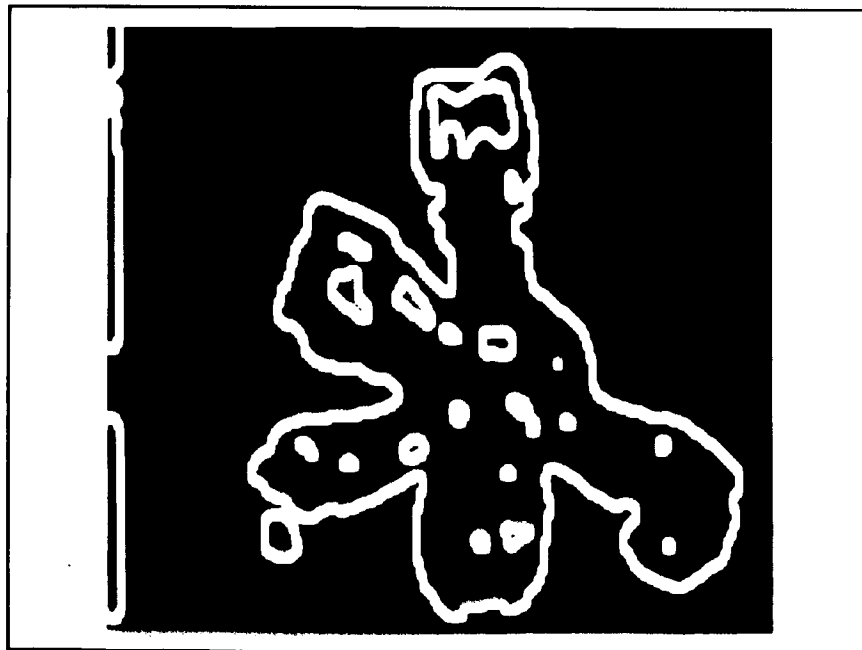


Figure 12. Sobel Edge Image

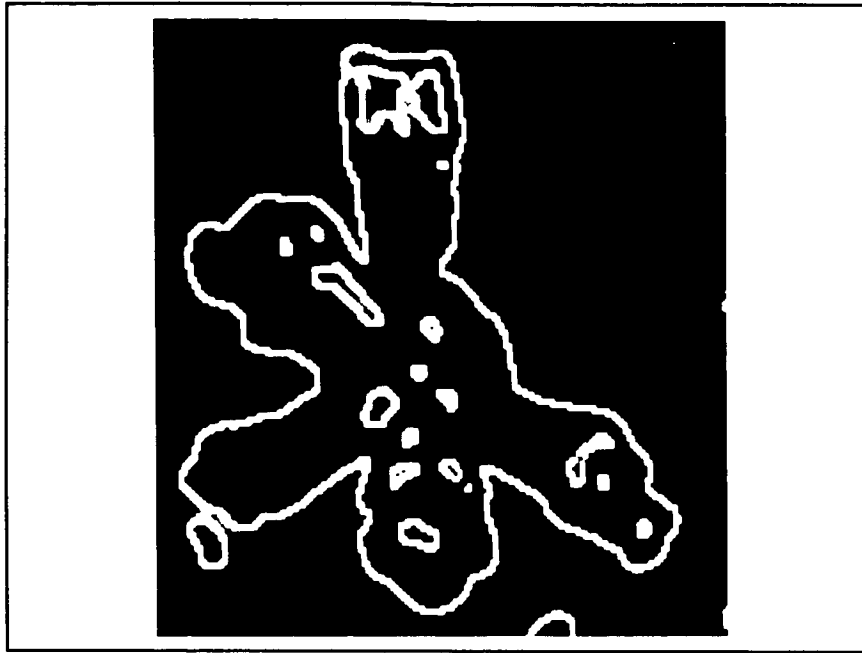


Figure 13. Thinned Edge Image

Using the hardware interface between PIPE and the HLP, the thinned edge image is transferred to the HLP for additional processing to obtain global information. In particular, the area of the node, its centroid, and its orientation are computed using the $(p+q)$ th order moments defined in [12]:

$$m_{pq} = \iint x^p y^q f(x,y) dx dy$$

where $f(x,y) = 1$ for all edge points and $f(x,y) = 0$ for all non-edge points. The centroid of an object is defined as :

$$x_c = m_{10} / m_{00}, y_c = m_{01} / m_{00}$$

where m_{00} is the area of the object, and the orientation is defined as :

$$\theta = .5 \tan^{-1} [2 (m_{00} m_{11} - m_{10} m_{01}) / ((m_{00} m_{20} - m_{10}^2) - (m_{00} m_{02} - m_{01}^2))]$$

The locations of corners of an object provide useful information in that they support the calculation of the orientation of an object. Given the model of an object, the viewing position can be determined by knowing which corners are visible.

Corners can be defined as locations where adjacent edge segments have high rates of curvature. These rates of curvature can be measured over small distances, yielding local corners. As the distance becomes larger, more global corners are found. To detect global corners, it is useful to use a lower resolution image, since a large area in the high resolution image maps to a relatively smaller area in the low resolution image (see Figure 10). A corner detection algorithm was implemented on the PIPE using these concepts.

Initially, an image of the truss node (see Figure 6) was used to generate successively lower resolution versions of the same image. The image was sampled so that only every other pixel on every other row was used to produce an image at the next resolution. From a 256 x 256 image, images were created of sizes 128 x 128, 64 x 64, 32 x 32, and 16 x 16. Using the low resolution image, a Sobel edge operator was applied to compute edge magnitude. Figure 14 is a picture of the edge image at this low resolution thresholded to indicate where edges resulted from high changes in contrast. Next, four Boolean neighborhood operations were computed on this binary edge image to test for the presence of eight types of corners (see Figure 15). The responses from the corner masks were combined and then expanded back to full resolution using pixel replication. The results are shown superimposed on the grey scale image of the truss node, where the corners were detected on the 16 x 16 level (see Figure 16) and on the 32 x 32 level (see Figure 17). As is expected, there are more responses obtained at the 32 x 32 level of resolution than at the 16 x 16 level. This is caused by the fact that the local operators are applied over a smaller distance, thereby detecting more local corners.

The quality and accuracy of the corners detected depend largely on the level of resolution at which they were extracted. The margin of error of the corner position is produced as a result of the way in which images are reduced and expanded on the PIPE. As an image buffer is reduced in resolution, pixels are sampled in every other row and in every other column. The result is always placed in the same corner of a 2 x 2 neighborhood. Expansion of an image involves the replication of pixels in a 2 x 2 neighborhood. Thus a corner point in the 16 x 16 image represents 16 pixels in the 256 x 256 image, any one of which can be a true corner point.

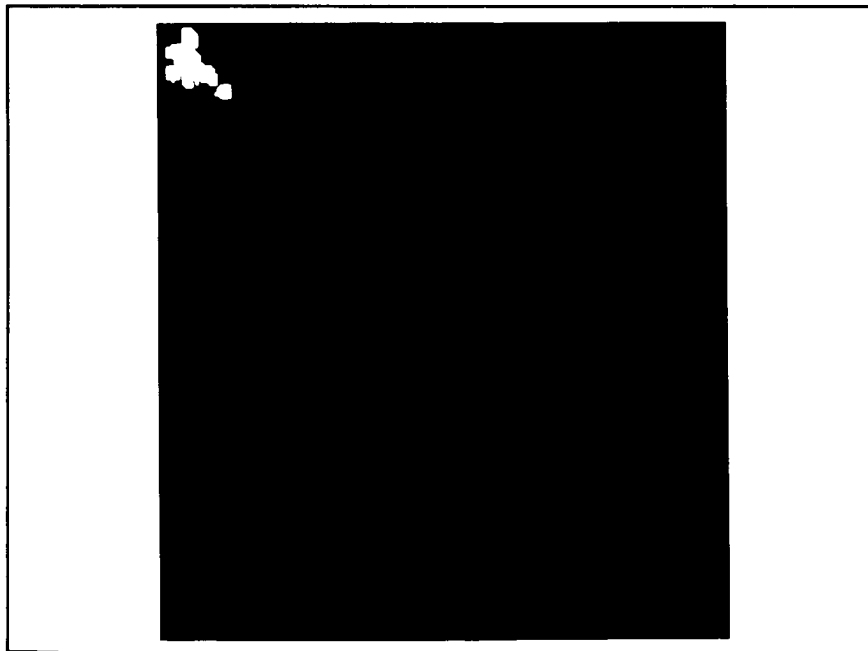


Figure 14. Low Resolution Image of Truss Node

Upper Left	Upper Right	Lower Right	Lower Left																																				
<table><tr><td>x</td><td>x</td><td>0</td></tr><tr><td>x</td><td>x</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>	x	x	0	x	x	0	0	0	0	<table><tr><td>0</td><td>x</td><td>x</td></tr><tr><td>0</td><td>x</td><td>x</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>	0	x	x	0	x	x	0	0	0	<table><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>x</td><td>x</td></tr><tr><td>0</td><td>x</td><td>x</td></tr></table>	0	0	0	0	x	x	0	x	x	<table><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>x</td><td>x</td><td>0</td></tr><tr><td>x</td><td>x</td><td>0</td></tr></table>	0	0	0	x	x	0	x	x	0
x	x	0																																					
x	x	0																																					
0	0	0																																					
0	x	x																																					
0	x	x																																					
0	0	0																																					
0	0	0																																					
0	x	x																																					
0	x	x																																					
0	0	0																																					
x	x	0																																					
x	x	0																																					
<table><tr><td>0</td><td>x</td><td>?</td></tr><tr><td>x</td><td>x</td><td>?</td></tr><tr><td>?</td><td>?</td><td>?</td></tr></table>	0	x	?	x	x	?	?	?	?	<table><tr><td>?</td><td>x</td><td>0</td></tr><tr><td>?</td><td>x</td><td>x</td></tr><tr><td>?</td><td>?</td><td>?</td></tr></table>	?	x	0	?	x	x	?	?	?	<table><tr><td>?</td><td>?</td><td>?</td></tr><tr><td>?</td><td>x</td><td>x</td></tr><tr><td>?</td><td>x</td><td>0</td></tr></table>	?	?	?	?	x	x	?	x	0	<table><tr><td>?</td><td>?</td><td>?</td></tr><tr><td>x</td><td>x</td><td>?</td></tr><tr><td>0</td><td>x</td><td>?</td></tr></table>	?	?	?	x	x	?	0	x	?
0	x	?																																					
x	x	?																																					
?	?	?																																					
?	x	0																																					
?	x	x																																					
?	?	?																																					
?	?	?																																					
?	x	x																																					
?	x	0																																					
?	?	?																																					
x	x	?																																					
0	x	?																																					

0 = no edge
x = edge
? = don't care

Figure 15. Corner Detection Masks

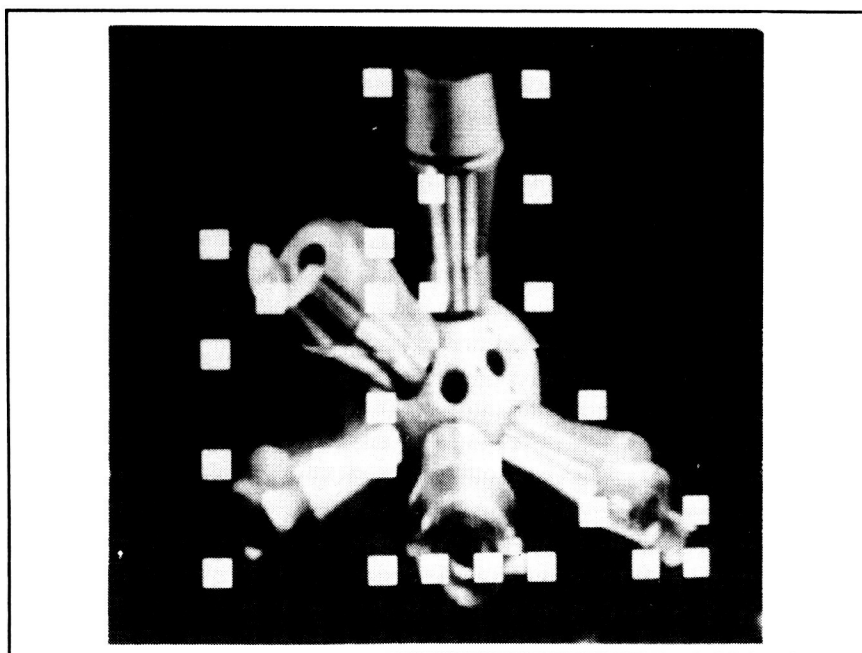


Figure 16. Corners Detected on 16 x 16 Level

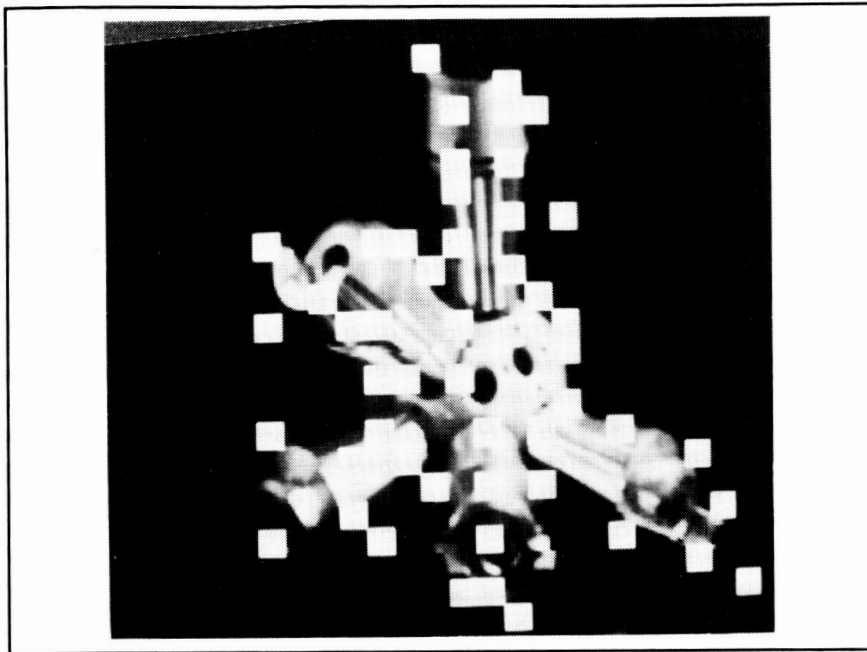


Figure 17. Corners Detected on 32 x 32 Level

5. Conclusion

A Flight Telerobotic Servicer will be used to assist the astronaut in the construction and maintenance of the NASA Space Station. NASREM, the hierarchical control system developed at the National Bureau of Standards (NBS), has been chosen by NASA as the computing architecture for this project. The sensory processing portion of this control scheme involves, at the low level, the preprocessing and enhancement of large arrays of data that have been gathered from external sensors. Feature extraction from these arrays is often more accurate if the data is preprocessed to remove the effects of noise and variable environmental conditions such as lighting.

Using a truss node, a structure which will be used in the Space Station, the PIPE was able to produce meaningful information which will be used by other processes in the control scheme. By using a combination of edge extraction techniques, image smoothing, and multi-resolution processing, image processing problems presented by the specularity of the truss node were overcome. A PIPE program is able to produce a thinned, connected edge image approximately every 1/10th of a second. A second PIPE program has been used to extract corners or areas of high curvature at update rates of 1/4th of a second. These results enable higher sensory levels to compute the position and orientation of the node in space.

More effort is required to bind corners detected at low levels of resolution with their true position in the full resolution image. This corner localization can be accomplished by adjusting the corner positions on each level of resolution during expansion instead of only at the lowest level. In addition, both algorithms can be enhanced by the removal of spurious edge points in the image.

Acknowledgement

The authors wish to thank Tsai-Hong Hong for her helpful suggestions and evaluations of algorithms implemented on the PIPE, and Brian Scafe for his photography.

References

- [1] Albus, J.S., McCain, H.G., Lumia, R., "NASA/NBS Standard Reference Model Telerobot Control System Architecture (NASREM)", NASA Document SS-GSFC-0027, December 4, 1986.
- [2] Arkin, R.C., Riseman, E.M., Hanson, A.R., "AuRA: An Architecture for Vision-Based Robot Navigation", Proceedings: Image Understanding Workshop, February, 1987.
- [3] Aspex, Inc., "PIPE--An Introduction to the PIPE System", 1987.
- [4] Fiala, J. , "Manipulator Servo Level Task Decomposition", Doc. ICG #002, NBS Internal Report, 1987.
- [5] Gross,T., Lam,M., Webb,J., "WARP As A Machine for Low Level Vision", IEEE Conference on Robotics and Automation, 1985.
- [6] Kent,E., Shneier,M., Lumia,R., "PIPE-Pipelined Image Processing Engine", Journal of Parallel and Distributed Computing, 1984.
- [7] Lumia, R., Shneier, M., Kent, E., "A Real-Time Iconic Image Processor", NBSIR, 1984.
- [8] Nashman, M., Chaconas, K., "Sensory Processing System, Data Acquisition Level", ICG #005, NBS Internal Report, 1987.
- [9] Nashman, M., Chaconas, K., "Sensory Processing System, Low Level Processing Stage", ICG #012, NBS Internal Report, 1988.
- [10] Rosenfeld, A., Kak, A., "Digital Picture Processing", Volume 1 Second Edition, Academic Press, 1982.
- [11] Wavering, A., "Manipulator Primitive Level Task Decomposition", ICG #003, NBS Internal Report, 1987.
- [12] Wilf, J.M., Cunningham, R.T., "Computing Region Moments from Boundry Representations", JPL Publication 79-49, November, 1979.

Autonomous Image Data Reduction by Analysis and Interpretation

Susan Eberlein, Gigi Yates and Niles Ritter

*Image Processing Applications and Development
Jet Propulsion Laboratory, Caltech 168-522
4800 Oak Grove Drive
Pasadena, CA 91109*

ABSTRACT

Image data is a critical component of the scientific information acquired by space missions. Compression of image data is required due to the limited bandwidth of the data transmission channel and limited memory space on the acquisition vehicle. This need becomes more pressing when dealing with multispectral data where each pixel may comprise 300 or more bytes. We are developing an autonomous, real time, on-board image analysis system for an exploratory vehicle such as a Mars Rover. The completed system will be capable of interpreting image data to produce reduced representations of the image, and of making decisions regarding the importance of data based on current scientific goals. Data from multiple sources, including stereo images, color images, and multispectral data, are fused into single image representations. Analysis techniques emphasize artificial neural networks. A stereogrammetry net reduces gray scale stereo images to a set of distance planes, indicating presence of objects at a given distance. Two stereo images, requiring one byte per pixel, can be reduced to eight image planes, represented in a single image of three bits per pixel. Images are divided into regions by combining distance data with edge locations, and each region described by its boundaries. Information on mineral composition of an image is derived from multispectral data. Spectra of each pixel are input to a classification neural net and a feature detector net; the output is the probable mineral composition of the region. This process reduces the 200-300 byte spectrum to a single mineral descriptor. The regions described above are subdivided until homogeneous in color and mineral class. Clusters are described by their outlines and class values. These analysis and compression techniques are coupled with decision making capacity for determining importance of each image region. Areas determined to be noise or uninteresting can be discarded in favor of more important areas. Thus limited resources for data storage and transmission are allocated to the most significant images.

PRECEDING PAGE BLANK NOT FILMED

INTRODUCTION

Over the next two decades, a number of scientific planetary missions are planned to land roving vehicles on planets and satellites, and to perform low orbit surveys. Image data plays two roles in achieving the goals of these missions. First, image data is used to make decisions about vehicle activities: stereo vision images are interpreted for use in navigation. Images from instruments such as an imaging spectrometer are used to determine which areas merit closer examination, and where physical samples should be taken.

Second, the images themselves contain significant scientific information. An imaging spectrometer collects reflectance data at multiple wavelengths for each pixel in an image. The resultant spectrum is characteristic for various minerals, so the multispectral image can be used to determine mineral composition of the area being surveyed. Image size can be enormous, since up to 300 wavelengths may be sampled for each image pixel. Compression of image data is required due to the limited bandwidth of the data channel and limited memory space on the acquisition vehicle.

This study develops methods for intelligent reduction of the image data obtained from stereo cameras and an imaging spectrometer, without loss of the most important scientific information. The scientific goals addressed by the prototype autonomous exploration system are those expected for a Mars rover: geological survey of Mars, searching for evidence of past water activity, volcanism, and fossil life.

We have developed a hierarchy of artificial neural networks and image analysis programs to autonomously extract information from the data of several sensors. High dimensional image data is reduced to a highly compact representation, containing the most relevant scientific information. This information may then be recorded or transmitted to earth, allowing the transmission of much more information than would be possible if uncompressed, unprocessed images were sent.

The current system consists entirely of software image analysis programs and software simulations of neural networks. Artificial neural networks are systems designed to emulate in some respects the functioning of biological neural systems (for overview and detailed descriptions see Rumlehart and McClelland, 1986). Information in a neural network is contained not only in memory locations or "nodes", but is distributed throughout the system in the weights of connections between the nodes. In a hardware network, these connections are the values of the electronic resistors, amplifiers, or inverters between two nodes.

Artificial neural networks have several advantages over traditional data processing techniques. They are robust when presented with noisy or incomplete input data. For example, a pattern matcher will find the closest match between a 20 dimensional input vector and a set of 20 dimensional vectors in memory, even when the input vector provides values for only 15 dimensions, with the other 5 values unknown. Additionally, neural nets lend themselves to hardware implementation in silicon microchips. We envision the construction of hardware neural nets to allow real time processing of images. Multiple nets may process pixels in parallel for maximum data throughput.

SYSTEM DESIGN

First, the overall flow of multisensor image data through the system will be outlined, then the function of each processing subsystem will be described in detail.

1. An edge finder takes as inputs two grey scale stereo images (each image consisting of P bytes, where P is the number of pixels). Edges are located at points where the slope between adjacent pixels is above a threshold. The output images have all pixels represented by "on" (edge present) or "off" (no edge).
2. A stereogrammetry net matches the two "edged" images to determine a distance for each pixel. The image is thus divided into a series of distance planes (labeled near to far), with each pixel assigned to a plane. If distance data is all that is required of an image, the stereo pair can now be represented by $P \cdot \log(n)$ bits, where n is the number of distance planes, reduced from the original $2 \cdot P \cdot 8$ bits.

3. The edge locations and the distance information are combined to outline contiguous regions or clusters.
4. For each cluster, sampling is begun with other instruments. In the simulation considered here, an imaging spectrometer samples a small subset of all available wavelengths (e.g. the color region) for several pixels in each cluster. If the sampled pixels are not homogeneous, the cluster is subdivided based on the location of the inhomogeneity, and resampled until homogeneous clusters are defined.
5. Each cluster of interest is sampled in a complete multispectral range. Subclustering of nonhomogeneous regions occurs until pixels from each outlined region fall into the same geological class.

After these steps are completed, the original images may be represented compactly as a set of regions. Each region is described by its edges and assigned information on distance from the viewing area, color, multispectral classification and any other appropriate sensor information. A compact representation of a region consists of arrays of triplets: line number, starting pixel, and ending pixel. A reconstruction program takes these arrays and redraws the image, interpolating line segments to produce the approximate outline, and attributing to each object in the image the multispectral classification derived from the original image.

The first step in the image analysis process, edge-finding, is not a data reduction technique in itself. However, it is necessary as a prerequisite to the region forming of step three, and can be used to speed stereo matching in step two. Edge finding is performed on two grey scale stereo images (Figure 1) using standard Sobel operator masks (see Gonzalez and Wintz, 1987). These masks calculate the slope in the vertical and horizontal directions in a 3 x 3 square centered at the pixel of interest (Figure 2). If the combined horizontal and vertical slopes are above a threshold, the pixel is designated an edge pixel (Figure 3A). Edge pixels are represented by a positive value in the output image, other pixels by zero. The edged images are used to speed initial stereo matching between corresponding pixels in the right and left stereo images, and as a first step in defining object outlines.

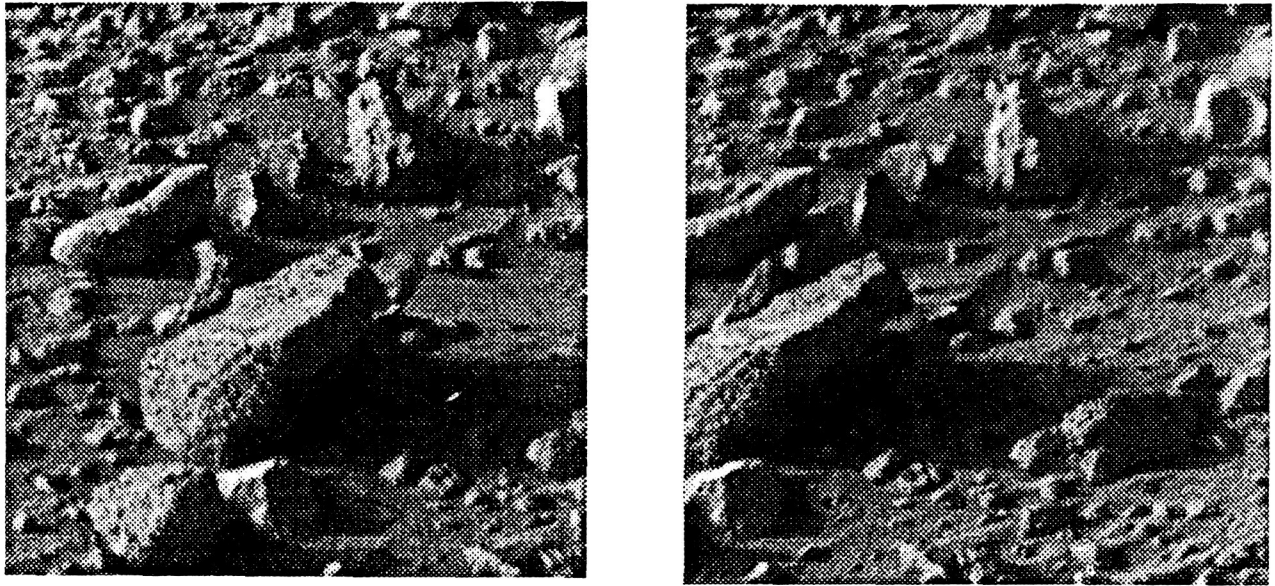


FIGURE 1. LEFT AND RIGHT STEREO IMAGES

-1	-2	-1
0	0	0
1	2	1

horizontal mask f

-1	0	1
-2	0	2
-1	0	1

vertical mask g

p1	p2	p3
p4	p5	p6
p7	p8	p9

pixel numbers

FIGURE 2. STANDARD SOBEL OPERATORS FOR EDGE FINDING

THE SLOPE VALUE FOR THE PIXEL LOCATED AT p5 IS $(f(p5)^2 + g(p5)^2)^{1/2}$ where:

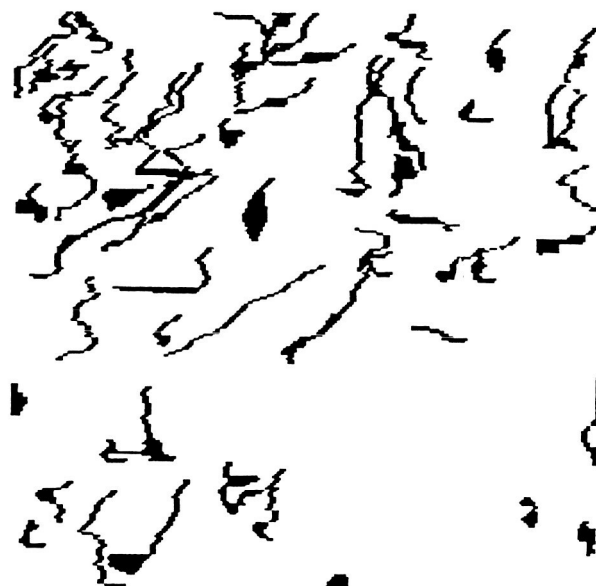
$$f(p5) = (p7 + 2p8 + p9) - (p1 + 2p2 + p3)$$

$$g(p5) = (p3 + 2p6 + p9) - (p1 + 2p4 + p7)$$

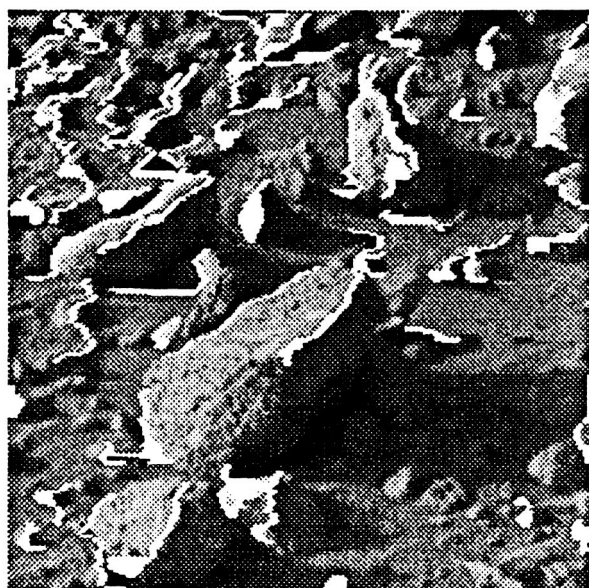
Potential edge pixels are followed and grouped into continuous edges. An edge following program starts with a single edge pixel then looks first to the left, then down, to the right and up until another edge pixel is located. Only adjacent pixels (horizontally, vertically, or diagonally) are considered. To be considered a complete edge, some minimum number of connected pixels must be found. This minimum threshold may be varied depending on the input image. Thresholding serves to eliminate isolated pixels and short edge segments that may derive from shadows or texture rather than true object edges. The edges shown in Figure 3B consist of segments with a minimum of 30 pixels. Figure 3C shows the positions of these edges on the original image.



A



B



C

FIGURE 3. EDGE FINDING

AFTER MASKING WITH SOBEL OPERATORS, ALL THE PIXELS WITH SLOPE VALUES ABOVE A USER DETERMINED THRESHOLD ARE ASSIGNED AN "ON" VALUE, AS SHOWN IN FIGURE 3A. ON PIXELS ARE THEN CONNECTED AND ISOLATED PIXELS REMOVED TO PRODUCE THE EDGES IN 3B. FIGURE 3C SHOWS THE OVERLAY OF THE ORIGINAL IMAGE AND THE EXTRACTED EDGES. NOTE THAT THE EDGES SHOWN IN 3B AND 3C HAVE BEEN WIDENED TO THREE PIXELS FOR BETTER DISPLAY.

The next step in image interpretation is to extract three dimensional information from the stereo pair (Figure 1) by stereo matching. For each pixel in the left image, a match is made with a "conjugate" pixel in the right image of similar intensity level. The horizontal offset between the two pixels is determined and used to approximate the distance to that point. Near objects have the greatest offset between corresponding points, far objects the least. The greatest problem in stereogrammetry is determining the correct conjugate for a pixel. In this case, a neural network simulation determines the matches, incorporating the intensity information and two additional constraints: uniqueness (a point exists at only one distance from the viewer) and continuity (adjacent points tend to be the same distance from the viewer) (see Marr and Poggio, 76).

The stereogrammetry net is based on a network developed by Sun et al. (1987), modified to process grey scale images. The net contains a "node" (a memory element which holds a numerical value between 0 and 1) for each pixel, in each of eight distance planes (Figure 4B). The net assigns each image pixel to one distance plane, based on the distance between the left and right conjugate pixels.

A model can be devised of the expected right-left pixel disparities for a standard scene where Rover "eyes" are looking out over flat terrain toward a horizon. Pixels near the top of the image will be more distant from the Rover, and therefore less separated between the right and left images (Figure 4A). Pixels near the bottom will be close to the Rover, thus showing more right-left disparity. The search for conjugate pixels starts with the assumption that matches will be made according to the flat terrain model. Any pixels that don't follow the expected distribution represent objects (such as rocks) disrupting the flatness. Such modeling will reduce the amount of time required to find conjugate points.

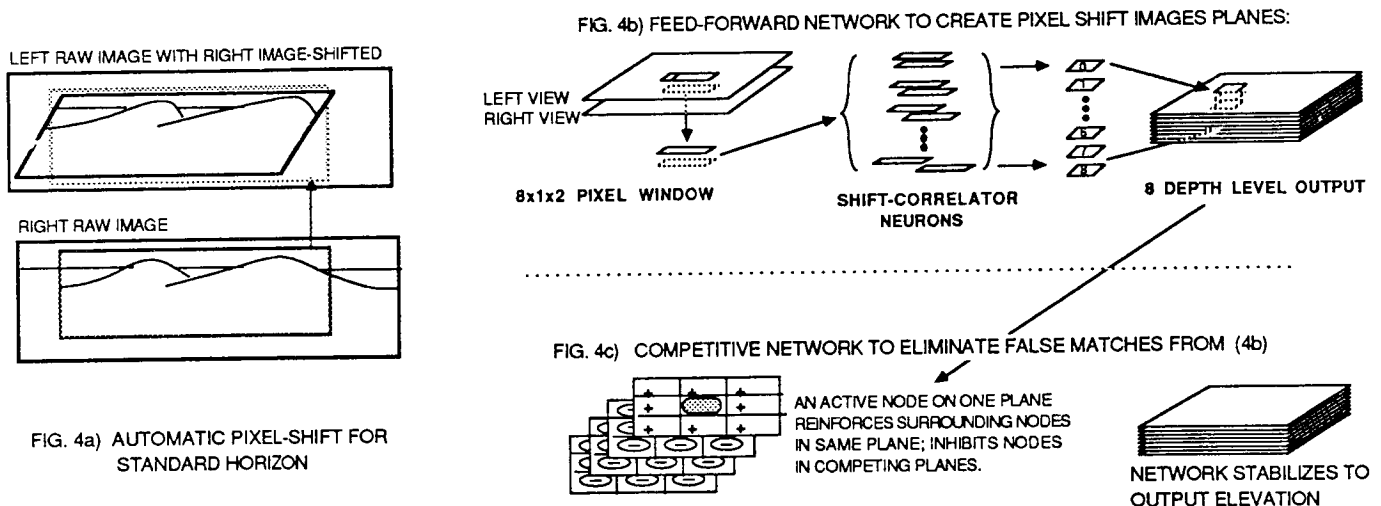


FIGURE 4. STEREOGRAMMETRY NEURAL NETWORK

Flat terrain modeling is the first step in stereo matching for the current prototype system. Dynamic modeling of the actual Rover images will require knowledge of the camera geometry and position on the rover, and the tilt angle of the vehicle.

For any pixel in the left image, there may be several prospective matches in the right image, based on pixel intensity level. If a pixel were matched simultaneously with several partners, it would imply that the point exists at several distances from the viewer, a physical impossibility. To circumvent this problem, inhibition occurs between a point in one distance plane and the corresponding point in every other distance plane (Figure 4C). The plane with the highest value for that point eventually inhibits the values in the other planes and forces them off. The result is that a particular point will be considered present (a value of 1.0 at the node) in only one

distance plane.

At the same time that inhibition occurs between the same points in different distance planes, reinforcement occurs between adjacent points in the same distance plane (Figure 4C). If a point has been turned on in several planes, this reinforcement allows the one distance plane to prevail over the others. The result is that a group of contiguous points will be placed together in the same distance plane, corresponding to the real world constraint that physical objects tend to be continuous. At edges of objects the inhibitory effects between planes will overrule the reinforcement.

The value contained in each node of the net is a function of the input value (positive when the right and left image pixels match at the distance corresponding to the current distance plane, zero otherwise), plus the sum of all the reinforcing node values minus the sum of all the inhibitory node values. The net goes through several iterations, calculating new values for each node until a stable state is reached. At the end, each point in the stereo image has been assigned to a distance plane. The distance information for the stereo pair may now be represented in $\log(n)$ bits per pixel, where n is the number of distance planes. For example, a stereo pair originally requiring two bytes per pixel can be divided into eight distance planes and described in three bits per pixel.

If distance information is all that is required from an image, processing may stop here. The stereogrammetry neural net has realized, in this example, a five-fold data size reduction while preserving the desired information. Traditional data compression techniques (such as run length encoding) may now be applied to the compact distance image. Alternatively, additional data may now be acquired from other scientific instruments and combined with the distance plane representation.

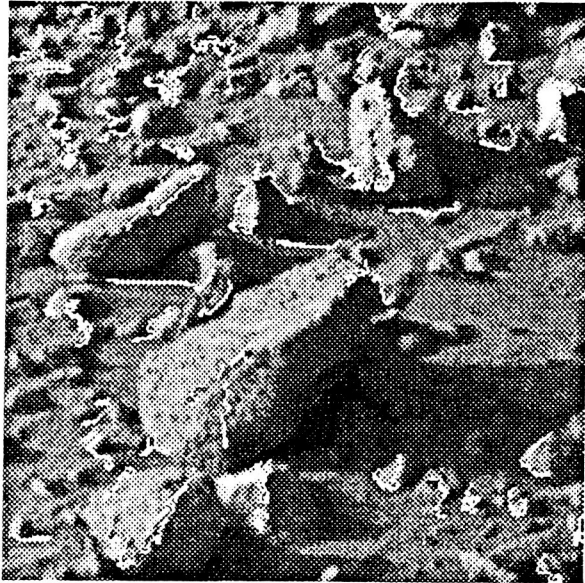
The edges located by the edge finder are combined with the distance information to define contiguous regions or clusters. Regions are defined in a single distance plane at a time, beginning with the nearest plane. The images used at this stage are the edged version of the LEFT stereo image and the distance plane representation, where each pixel in the image is assigned a value of "on" or "off" (Figure 5B). The left stereo image is required for the edges (Figure 5A), since this is the reference image in the stereo matching. That is, the pixel location in the left image will correlate with the point assigned a distance in the stereo matching output.

Regions are defined in each distance plane by starting at the leftmost boundary of the plane and looking to the right, up, and down, until an edge is encountered, or until the pixels no longer fall in the current distance plane (Figure 5B). A region thus outlined is described by its boundaries, and the enclosed pixels are eliminated from further consideration at this step. The region definition step repeats until all the pixels in the distance plane have been assigned to regions (Figure 5C). Note that the regions do not exactly correspond to physical objects. Some regions may comprise more than one object where there were no well defined edges; others may be subdivisions of objects due to shadows. However, the rough division of the image into sections allows sampling with scientific imaging instruments to proceed without examining every pixel in the image.

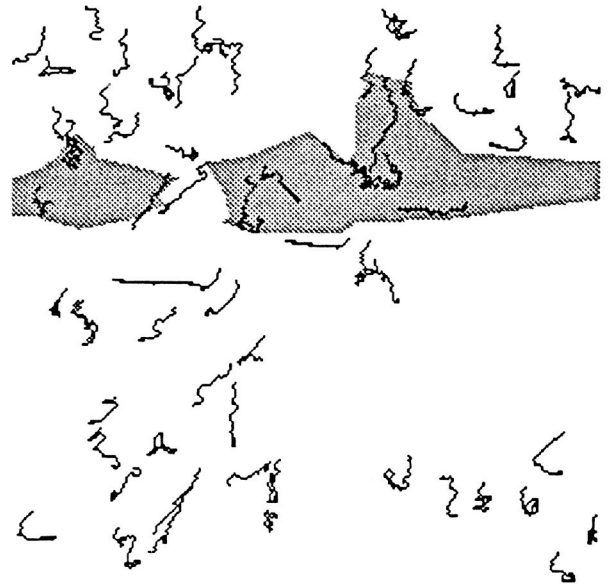
Once regions are defined, data is collected with the scientific sensing instruments for each region. An imaging spectrometer will sense and record the light intensity at the desired wavelengths for the entire image frame. However, analysis of this very large data set begins with only a subset of the recorded pixels. The set of pixels sampled may be chosen at random, or may be chosen to subdivide the region geometrically to ensure that samples encompass the entire region. A simple scheme of geometric subdivision is to require that any region with dimensions greater than, say, 10 pixels, be divided into quarters and pixels analyzed from each subregion. The resulting regions are necessarily approximations to the actual physical objects, but are sufficient for purposes of scientific imaging.

Here we describe the analysis of data acquired with an imaging spectrometer. At first, only a small subset of all the available wavelengths are sampled, for example the visible color wavelengths. Two approaches are available for determining whether the region is homogeneous in color:

In the first approach, the average color is determined and distance to that average calculated for each pixel. If there are three input values being considered (e.g. red, green and blue), then each pixel is represented by a three dimensional vector. The distance between the color value for a pixel and the average for a region is the L1 distance between the vectors. If the distance is too great, subclusters are formed to separate the non-homogeneous regions, and the procedure is repeated.



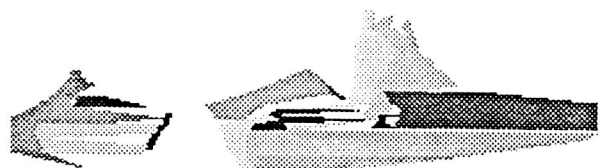
A



B



C



D

FIGURE 5. REGION DEFINITION FROM EDGE AND DISTANCE DATA

THE EDGES ARE SUPERIMPOSED ON THE ORIGINAL IMAGE IN FIGURE 5A, AND ON THE DISTANCE PLANE TO BE CONSIDERED IN FIGURE 5B. NOTE THAT THE DISTANCE PLANE IS A SIMULATION OF THE EXPECTED OUTPUT OF THE STEREOGRAMMETRY NET. FIGURES 5C AND 5D SHOW THE OUTPUT OF THE CURRENT REGION OUTLINING PROGRAM. 5C IS THE ORIGINAL OUTPUT, REQUIRING 662 BYTES. THE IMAGE IN 5D IS RECONSTRUCTED FROM THE COMPACT DESCRIPTION FORMAT, REQUIRING 312 BYTES. THE DIFFERENT GREY LEVELS IN THE IMAGES REPRESENT DIFFERENT OBJECTS OR REGIONS.

The second approach requires that a list of expected color combinations be determined in advance. These colors are stored as memories in a neural net, using a grandmother cell classifier architecture (Figure 6). The color values for each pixel are presented to the net, and the pixel is assigned to the closest color class and given a certainty value to indicate how good a classification was made. If all pixels fall into the same class with high enough certainty, the clustering is finished. Otherwise, the region is subdivided to separate pixels in the different classes.

The second approach has the advantage that the different color classes can be assigned "interest" levels, depending on the current goals of the overall system. For example, if the goal of the system is to identify any minerals which contain iron, the color classes of interest would be those with a high red component. Any regions which do not fall in the appropriate color classes can be eliminated from further consideration, and not even be recorded in the final compact representation of the image data. The current prototype system employs this approach.

It should be noted here that the choice of the color region for preliminary sampling is purely for demonstration purposes. In the final system, other wavelengths out of the visible range may have greater importance for classifying minerals and distinguishing the important and unimportant images. The final choice of wavelengths for the first sampling step will depend on the expected distribution of minerals.

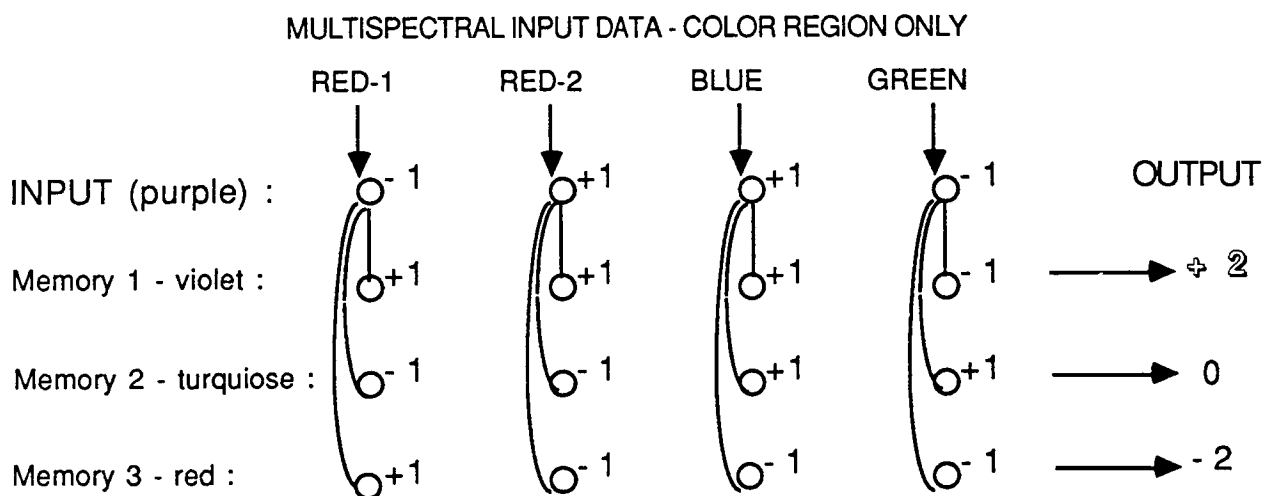


FIGURE 6. NEURAL NETWORK CLASSIFIER BUILT WITH GRANDMOTHER CELL ARCHITECTURE

INPUT VECTOR DERIVES FROM MULTISPECTRAL DATA AT SEVERAL WAVELENGTHS. THE EXAMPLE SHOWN HERE IS A SET OF FOUR INPUT VALUES TAKEN FROM THE COLOR REGION OF THE VISIBLE LIGHT SPECTRUM. THE INPUT IS MATCHED TO THE CLOSEST OF SEVERAL MEMORY VECTORS (EACH MEMORY REPRESENTING ONE COLOR) BY TAKING THE DOT PRODUCT. THE MEMORY WITH THE HIGHEST OUTPUT VALUE IS CLOSEST TO THE INPUT VECTOR. THE VALUES SHOWN HERE ARE +1 OR -1 FOR EACH NODE, BUT MAY BE REPLACE BY ANALOG VALUES FROM 0 TO 1.

After preliminary sampling and discarding "uninteresting" regions, each cluster of interest is sampled in a complete multispectral range. Pixels are chosen in the same manner as for color sampling. At this step, only the neural net classifier approach is used to determine whether pixels are homogeneous. The same grandmother cell architecture as shown in Figure 6 is used for classifying complete spectra, but the number of input values for each pixel is increased from three to the total number of wavelengths sampled. In the current classifier network simulation, 32 values over the wavelengths from 2.0 to 2.5 micrometers are sampled. These are classified into 15 groups, where a typical spectrum of each geological mineral type (e.g. carbonates, clays) is the memory for the class. The final system will classify up to 300 input values (over the wavelengths from 0.3 to 5.0 micrometers)

into perhaps 40 classes, including more specific mineral types and mineral mixtures.

As in the color case, two decisions are made as a result of the multispectral data classification. First, sub-clustering of nonhomogeneous regions occurs until all the pixels in each region fall into a single geological class. Second, those regions composed of uninteresting classes are eliminated from further analysis.

The goal of the preceding steps is to generate a compact description of each region of interest. During each clustering step (i.e. region forming from edges and distance planes, color sampling and classifying, multispectral sampling and classifying) two outputs are produced: a temporary image containing the information derived from that step, and an array describing each cluster under consideration. The arrays have classification information (for example, distance plane 4, color class 12, mineral class 14) and then a list of starting and ending pixels for each line in the region. The classification information can be represented in one or a few bytes, while the original imaging spectrometer data required up to 300 bytes.

After sufficiently homogeneous clusters have been defined, this array format is automatically reduced to an array of triplets. Each triplet contains line number, starting pixel, and ending pixel. However, not every line in the object needs to be recorded. The current array reduction program chooses lines at a constant interval, the interval size depending only on overall object size. For example, an object of 100 lines may have the parameters recorded for every sixth to tenth line. A reconstruction program then interpolates straight line segments between the points of the recorded lines to reconstruct the rough outline of the object.

Figure 5C is the image described by the original array and 5D is the image reconstructed from the reduced array form. The region descriptors at the first step required 662 bytes to completely describe the outline; 312 bytes are required after the descriptive arrays are reduced to the most compact form. This is derived from an original image of 250 x 250 pixels. Since the distance plane in Figure 5B contains only about one sixth of the total 62,500 pixels, it is expected that the entire image could be represented with on the order of 1800 bytes. This representation includes not only the object shapes, but also their distance, color and mineral classification.

A future refinement of the array reduction algorithm will examine the slope of the actual object boundary and choose the lines to record so as to fit the reconstructed object shape more closely to the original object shape. For certain applications (e.g. vision for path planning of a roving vehicle), good object reconstruction is a requirement. For other applications (e.g. determining the level of geological diversity for various areas in the field of view), the rough approximation to object shape is adequate.

Most of the multispectral imaging on a Mars Rover will be oriented toward geological survey of a region and determining the interest level of the minerals present. For these purposes, the location and approximate size and shape of each geologically distinct region is sufficient information. Thus very compact representations can be produced, realizing data size reductions of three orders of magnitude from the original visible images. Because the mineral composition information is in the form of a classification value rather than a several hundred byte spectrum, the total reduction in data size is on the order of five to six orders of magnitude.

CONCLUSIONS

Intelligent methods of image data reduction are required in order to acquire maximum information from a planetary exploration mission such as the Mars Rover mission. Image data sets, particularly data acquired over multiple wavelengths from an imaging spectrometer, can be enormous. Much of this data is repetitive (as when a large rock or plain is homogeneous in mineral composition) or not sufficiently important to record or transmit. If a Rover is to transmit more than a few images in one communication window, dramatic reductions in image size are required, while still preserving the scientific content of the data.

We have developed a software prototype of a system which exploits the redundancy and variable information content of visible and multispectral images to realize large reductions in data size. Information on distance from the viewer, object color and mineral content is extracted and represented in compact form. The information that is preserved is chosen to fit the current scientific objectives of the exploratory mission. Homogeneous regions in the images are represented by compact descriptions of their outlines, rather than a pixel by pixel description. This approach is not suited to some imaging applications, such as fine scale navigation, but it is well suited to geological surveying and the types of applications expected of an imaging spectrometer on the Mars Rover.

The software simulations of the prototype system demonstrate that intelligent reduction of image data by five to six orders of magnitude is feasible. Development of dedicated hardware for some of the analysis steps, and of neural network chips, will allow real time analysis and reduction of high dimensional image data.

REFERENCES

Gonzalez, R., and Wintz, P. (1987). Digital Image Processing. Addison-Wesley Publishing Company Inc., Reading, MA.

Marr, D., and Poggio, T. (1976). Cooperative Computation of Stereo Disparity. *Science* 194: 283-287.

Rumelhart, D., and McClelland, J. (1986). Parallel Distributed Processing. MIT Press, Cambridge, MA.

Sun, G. Z., Chen, H. H., and Lee, Y. C. (1987). Transactions of the IEEE First International Conference on Neural Networks, vol. 4: 345-355.

AN AUTOMATED COMPUTERIZED VISION TECHNIQUE FOR DETERMINATION OF THREE-DIMENSIONAL OBJECT GEOMETRY

Pen-Tai Chiang * Jackson C. S. Yang[†] and V. Pavlin[‡]

Robotics Laboratory
Department of Mechanical Engineering
University of Maryland
College Park, Maryland 20742
TEL: (301) 454-7694

April 25, 1988

ABSTRACT

It is very important to determine three dimensional geometry of objects quickly in various military, space, construction and industrial applications. This paper presents an automatic scheme to obtain three dimensional geometry of objects by employing only one camera. At present, this technique is applicable to a limited category of objects, satisfying the following constraints: they are flat-surfaced, and all the vertex points have to be recognized as corner points of the two dimensional image. The scheme consists of corner detection, data communication, camera calibration techniques and point searching and matching, edge cancelation and creation procedures.

An "L" shaped model is chosen as a test object. Experimental results demonstrated the reconstruction of this object geometry within 5 mm discrepancy. This scheme is quite convenient, efficient to use and can be applied to a wide range of problems in the real world.

*Research Assistant

[†]Director, Robotics Laboratory

[‡]Research Associate

1 INTRODUCTION

It is very important to determine three dimensional geometry of objects quickly in various military, space, construction and industrial applications. Among the techniques used to achieve this goal is the Noncontact Measurement Technique (NMT). This technique uses a vision system which involves photographic projection, digitalization and computerized computation.

Two schemes have been developed to implement the concept of NMT. The first one is the structure light approach [1]. With structured light shining on an object, this approach analyzes the two dimensional image. Based on the light intensity of the reflection, the object's edges can be recongnized. However, this technique requires a harmonious working environment and expensive equipment. Another disdavantage is that only the qualitative descprition of the object is available, and quantitive information is not easily retrieved. The second approach is to use an optoelectronic device, such as the commercial package Selslpot [2], to detect Light Emitted Diodes (LED), which are attached to an object at the various points of interest. The 3D coordinates of these LED can be calculated with two cameras. This is a very time consuming process, and the problem with the LED being obstructed from the cameras places a limit to this technique in 3D measurements. The motivation of this project is to develop an advanced computerized vision technique for automated determination of three dimensional object geometry.

The main goal of this paper is to obtain the three dimensional geometry of an object by employing only one electronic camera. The camera will be moved around the object, and take pictures from different location and orientation. At present, this technique is applicable to a limited category of objects, satisfying the following constraints: they are flat-surfaced, and all the vertex points have to be recognized

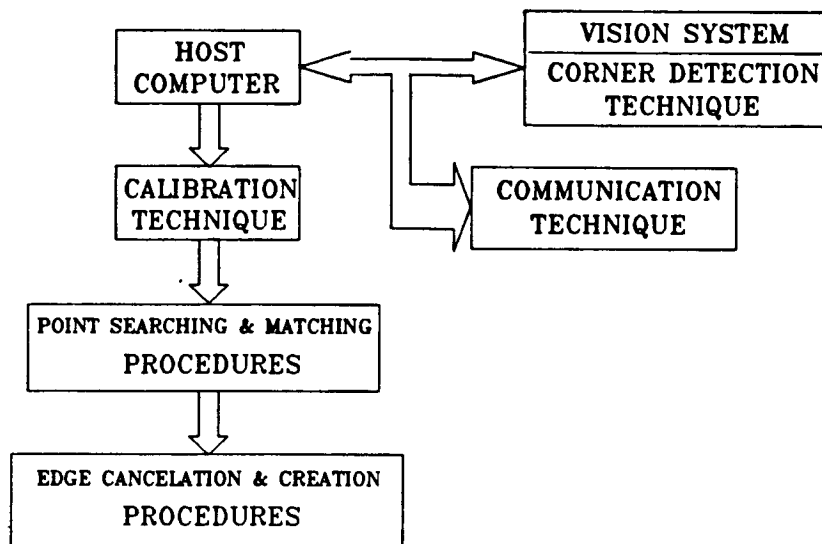


Figure 1: Flow Chart of the Proposed Scheme

as corner points of the 2D image. Finally, a graphical wireframe representation of the object will be presented.

2 GENERAL SCHEME

The proposed scheme is developed in a general way such that it could be adapted in many vision systems. The overall structure of this scheme is illustrated in fig. 1. Essentially, five modules are developed and incorporated. The purpose and function of each module are described in the following.

1. Corner Detection Technique

The 2D image of an object satisfying the previously defined constraints will always be a polygon. Each corner point of this polygon represents the projection image of an object's vertex point. In order to obtain the 3D coordinates of the vertex points, the 2D coordinates of all the corner points have to be extracted first. The basic philosophy of corner detection technique adapted is a combination of classic and fuzzy model approach [3]. In addition, each boundary section of the polygon

represents the projection image of an object's edge. Edges are very important to graphically represent the object. The sequencing of the detected corner points along the polygon should be either in a clockwise or counterclockwise direction. Then by connecting two successive corner points an edge can be created between their corresponding vertex points.

2. Data Communication Technique

A host computer and a vision system are the two major pieces of equipment involved in the proposed scheme. The controlling signal from the host computer to the vision system and the 2D coordinates from the vision system to the host computer will be transmitted back and forth. All the necessarily transmitted information is converted into ASCII code in the host computer through its communication port. In the case of the controlling signal, one ASCII code is capable of doing the job. For example: "0" is to "close", and "1" is to "open". For the 2D coordinate data, a transformation program is developed to carry out the conversion of a series of ASCII code into real value data.

3. Camera Calibration Technique

Since the camera of the vision system is moved and arbitrarily placed around the object of interest, the location and orientation of the camera has to be calibrated with respect to a predefined Reference Coordinate System (RCS) (X, Y, Z) (fig. 2), before the measuring process commences.

Referring to the study of [2] [4] [5], 3D coordinates of some points in the RCS and their corresponding 2D coordinate in the Image Coordinate System (ICS) (h, v) have to be known to calibrate the camera. The equation showing the transformation relation between point $S (\hat{X}, \hat{Y}, \hat{Z})$ in RCS and its corresponding projection $S' (\hat{h},$

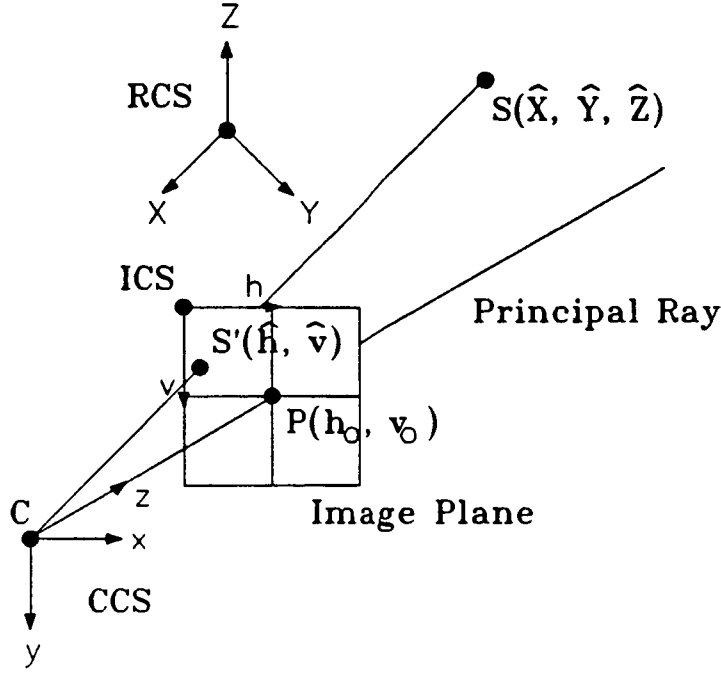


Figure 2: Schematic Diagram of Various Coordinate Systems and Projection Model

\hat{v}) in ICS is:

$$S' = \begin{pmatrix} \hat{h} \\ \hat{v} \\ 1 \end{pmatrix} = ((R)_{3 \times 3}, (T)_{3 \times 1}) * \begin{pmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \end{pmatrix} = [CAM] * S \quad (1)$$

where

$(R)_{3 \times 3}$ is the orthogonal rotation matrix of camera's orientation

Camera Coordinate System (CCS) with respect to RCS.

$(T)_{3 \times 1}$ is the camera's location with respect to RCS.

$[CAM]$ is called the camera transform.

It has been shown [4] that $[CAM]$ is uniquely determined if at least four pairs of points $[(\hat{X}_i, \hat{Y}_i, \hat{Z}_i); (\hat{h}_i, \hat{v}_i)]$ are known.

4. Point Searching And Matching Procedures

For a given 2D image, the unit vector (\vec{n}) directing from the camera center C to the principal point P on the image plane is known as $(R_{3 \ 1}, R_{3 \ 2}, R_{3 \ 3})$ with respect

to RCS after calibration process. Provided that the center of the camera view (\hat{h}_0, \hat{v}_0) in ICS and focal length f in CCS are given, the unit vector (\vec{n}_i) directing from the camera center to any corner point (\hat{h}_i, \hat{v}_i) can be calculated by the following equation:

$$(\vec{n}_i) = \begin{pmatrix} \hat{h}_i - \hat{h}_0 \\ \hat{v}_i - \hat{v}_0 \\ f \end{pmatrix}^T ((R)_{3 \times 3}) \quad (2)$$

Suppose that m pictures are taken, and for the i^{th} picture there are n_i corner points, a total of $\sum_{j=1}^m n_j$ unit vectors are obtained. If multiple images of a point in space is obtained, the extension of its corresponding unit vectors should intersect at this point. A searching procedure is developed to go through all the unit vectors to find those that intersect. Since the inaccuracy of corner detection will cause inaccuracy in the calculation of a unit vector, two unit vectors are considered to intersect if their offset distance is less than a preselected tolerance. The largest number (should be less than or equal to m) of intersecting unit vectors are grouped together first. This process is continued until the number of intersection of unit vectors decreases to two. Based on trigonometry, the optimally matched 3D coordinates are identified to represent the vertex point of the object. All the vertex points can then be identified.

5. Edge Cancellation and Creation Procedures

By connecting two successive corner points an edge can be created between their corresponding vertex points. It is quite possible that a redundant edge appears. This happens when the camera's orientation is nearly parallel to one or more boundary planes of the object. A checking program is developed to cancel the redundant edges. In some cases, a physically existing edge can not be recovered because it is not shown as a boundary section of any image pictures. A special program is coded to recover these kind of edges.

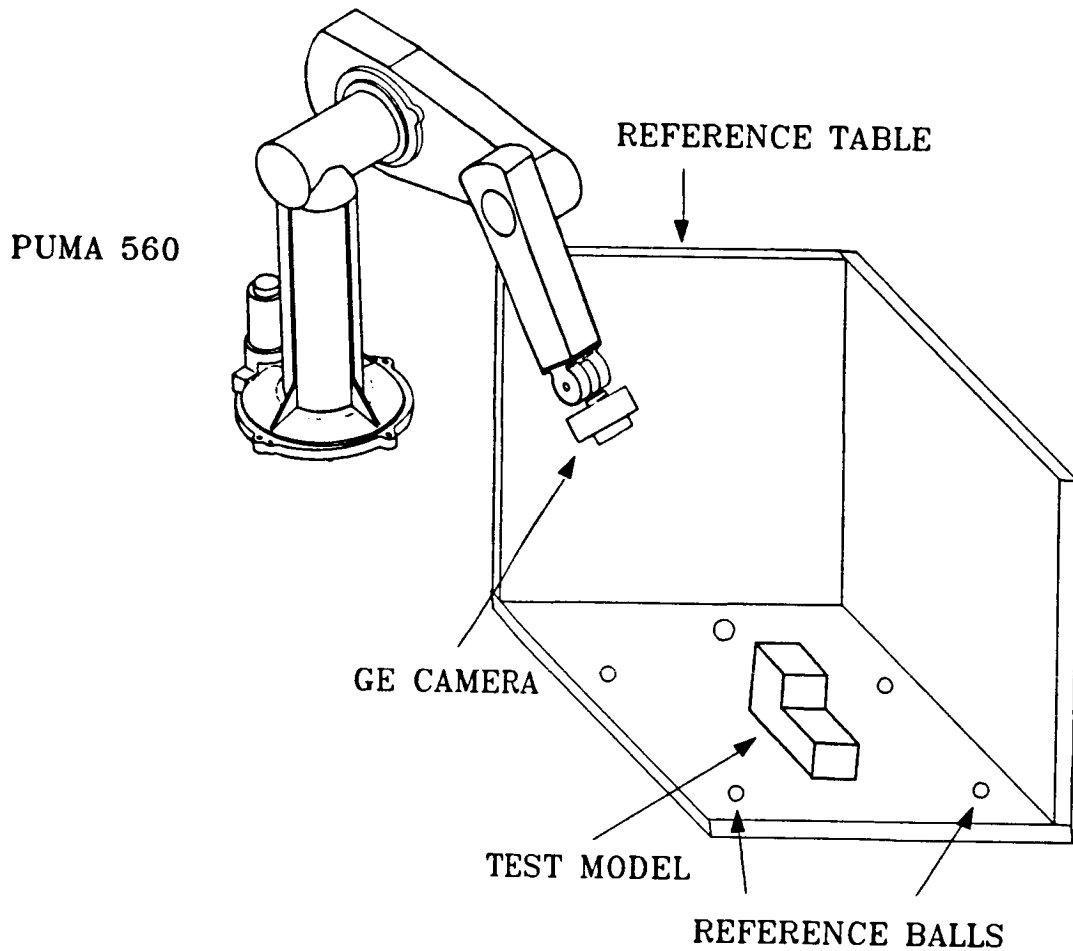


Figure 3: Experimental Set Up

3 EXPERIMENTAL SET UP AND RESULTS

Experimental Set Up

In order to implement the proposed scheme in a fully automatic sense, a camera is attached to a robot arm. This experimental set up (fig. 3) includes an IBM-PC with two series communication ports, GE vision system, PUMA 560 robot arm, a reference table and a test model.

(1) Reference Table:

The reference table consists of three pieces of black board, orthogonal to each other to represent the quadrant of a RCS. Five reference balls are glued on the base

NO.	CENTRAL COORDINATE			RADIUS (mm)
	X (mm)	Y (mm)	Z (mm)	
1	50.0	0.0	12.0	24.0
2	0.0	200.0	6.0	12.0
3	250.0	250.0	6.0	12.0
4	300.0	100.0	6.0	12.0
5	100.0	300.0	6.0	12.0

Table 1: Central Positions and Radii of Reference Balls

NO.	CENTRAL COORDINATE			RADIUS (mm)
	X (mm)	Y (mm)	Z (mm)	
1	153.0	90.0	102.0	24.0
2	100.0	220.0	0.0	12.0
3	153.0	220.0	0.0	12.0
4	100.0	90.0	102.0	12.0
5	153.0	90.0	0.0	12.0
6	100.0	220.0	53.0	12.0
7	100.0	150.0	102.0	12.0
8	153.0	220.0	53.0	12.0
9	100.0	150.0	53.0	12.0
10	153.0	150.0	102.0	12.0
11	153.0	150.0	53.0	12.0
12	100.0	90.0	0.0	12.0

Table 2: Vertex Points' Coordinates of the Test Model

board at predetermined positions. One of these balls is larger than the other four. The reason for this special arrangement will be explained later. Their radii and central positions are tabulated in table 1.

(2) Test Model:

The test model is white, and "L" shaped, whose dimensions are illustrated in fig. 4. The coordinates of the model's vertex points with respect to the coordinate system set up by the reference table are tabulated in table 2. Its white appearance is to enhance the intensity of light contradiction with the black reference table. This is helpful in the detection of corner points.

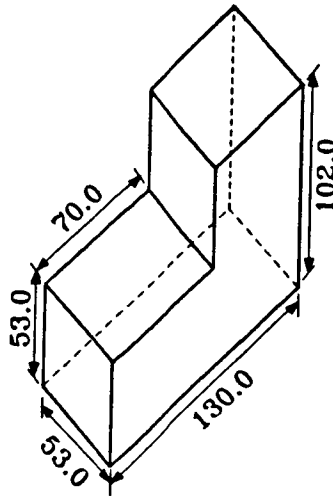


Figure 4: Dimension of the Test Model

(3) PUMA 560 Robot Arm:

A GE camera is attached to the wrist of a PUMA 560 robot, which moves to various location upon receiving a "motion" signal from its external I/O port. Once reached the location, the controller will send a "ready" signal to the host computer through one of its communication ports.

(4) GE Vision System:

The GE vision system consists of an electric camera and system controller. The camera will take a picture of the model upon receiving a "taking a picture" signal from the controller's external I/O port. A typical two dimensional image is shown in fig. 5. After detecting the corner points and extracting the centers of the five reference balls from the images, the transmitting sequences of these 2D coordinates data through the port are as follows: Center of the largest reference ball, centers of the other four reference balls in the order from top to bottom, highest corner point in the model's image, and then the remainder of all the detected corner points in a counterclockwise sequence. Finally, a "ready" signal is sent to the host computer through the other communication port indicating the end of data transmission.

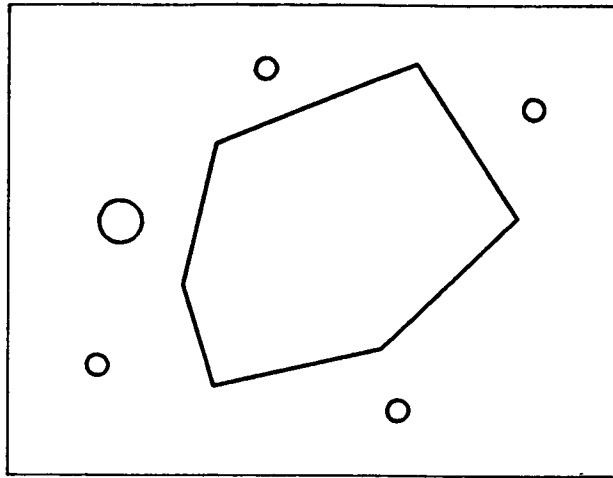


Figure 5: Example of 2D Image

(5) IBM-PC:

An IBM-PC is used as the host controller in the experimental system set up. Two communication ports are connected to the vision system and the robot arm controller respectively. A software (fig. 6) is embedded to maneuver the operating sequences and to obtain the 3D coordinates of the model. The process of moving the robot arm and taking pictures in step 1 will be repeated until enough 2D image coordinate data have been extracted from images taken from different angles and locations. Based on these data, the analyzing procedures to calibrate the camera, search and match corner points, cancel and create edges of a model are executed in step 2. The identified edges and corner points of the model and their 3D graphical wireframe representation are provided in step 3. This whole process is fully automated.

Experimental Result

Since PUMA 560 robot arm's working volume is limited, the camera attached to the robot is not able to provide sufficient coverage of the model from different angles and locations to achieve the measurement process. Some of the images are obtained by moving the camera around the model by hand and taking pictures.

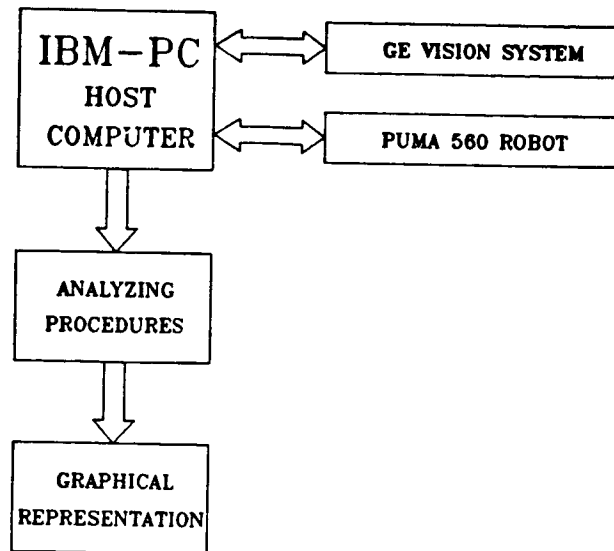


Figure 6: Flow Chart of Experimental Set Up

This picture taking process is repeated until all the vertex points of the model are shown to be the corner points of an image at least twice. All the necessary 2D coordinate data are analyzed and stored at the specific file structured as shown in fig. 7. A "taking a picture" and "ready" signal are inserted into the file at the beginning and end respectively. The vision system will transmit only portion of the file upon receiving a "taking a picture" signal, and terminate after sending a "ready" signal. The analyzing process will start when all the data contained in the file are transmitted.

In this experiment, ten pictures are taken. Though this is not necessary, the purpose is to compensate for the inaccuracy of the vision system and to enhance the reliability of the results. The resulting vertex points' coordinates and edge connection sequences of the model are tabulated in table 3 and table 4 respectively. The graphical wireframe representation of the model, dumped from the IBM-PC screen, is shown in fig. 8.

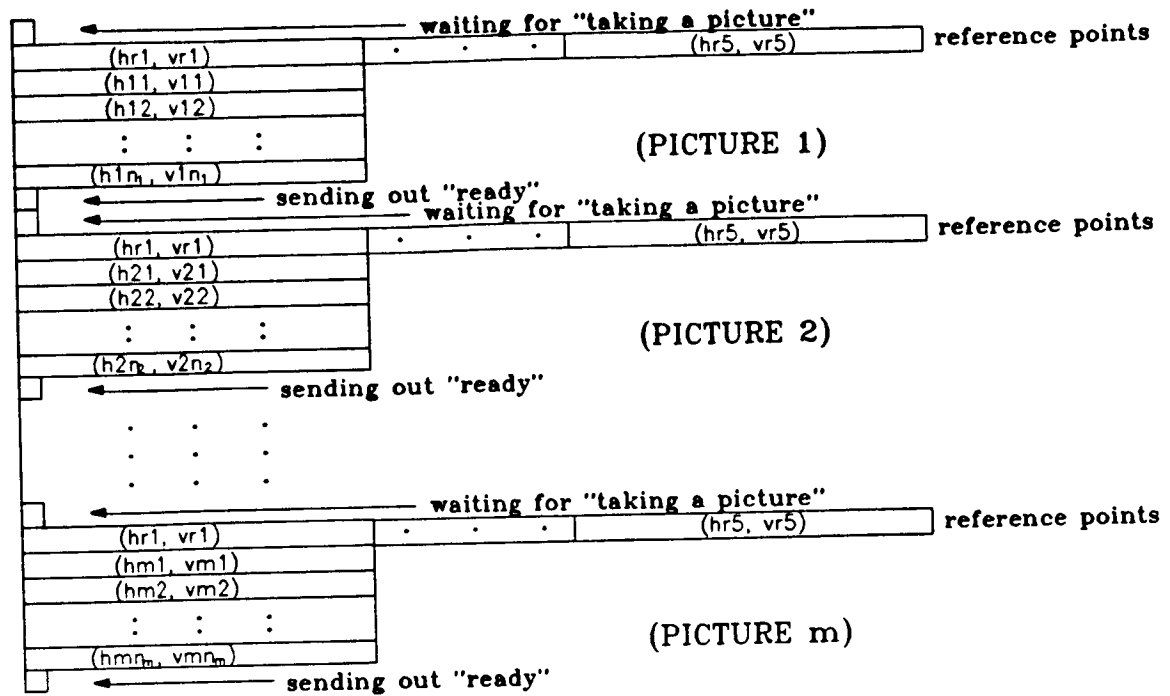


Figure 7: Data Structure of the Vision File

	X (mm)	Y (mm)	Z (mm)		X (mm)	Y (mm)	Z (mm)
1	151.0	92.0	104.0	7	95.0	150.0	96.0
	(153.0	90.0	102.0)		(100.0	149.0	102.0)
2	102.0	221.0	0.0	8	153.0	220.0	50.0
	(100.0	220.0	0.0)		(153.0	220.0	53.0)
3	154.0	221.0	0.0	9	102.0	150.0	56.0
	(153.0	220.0	0.0)		(100.0	154.0	53.0)
4	99.0	95.0	103.0	10	154.0	150.0	98.0
	(100.0	90.0	102.0)		(153.0	149.0	102.0)
5	153.0	92.0	1.0	11	1510	150.0	55.0
	(153.0	90.0	0.0)		(153.0	152.0	53.0)
6	101.0	221.0	49.0	12	102.0	89.0	2.0
	(100.0	220.0	53.0)		(100.0	90.0	0.0)

Table 3: Obtained Vortex Points' Coordinates (Compared with the Actual Values in Parentheses)

EDGE NO	CONNECTION	EDGE NO	CONNECTION	EDGE NO	CONNECTION
1	1 - 4	7	3 - 5	13	6 - 9
2	1 - 5	8	3 - 8	14	7 - 9
3	1 - 10	9	4 - 7	15	7 - 10
4	2 - 3	10	4 - 12	16	8 - 11
5	2 - 6	11	5 - 12	17	9 - 11
6	2 - 12	12	6 - 8	18	10 - 11

Table 4: Edge Connection Sequence

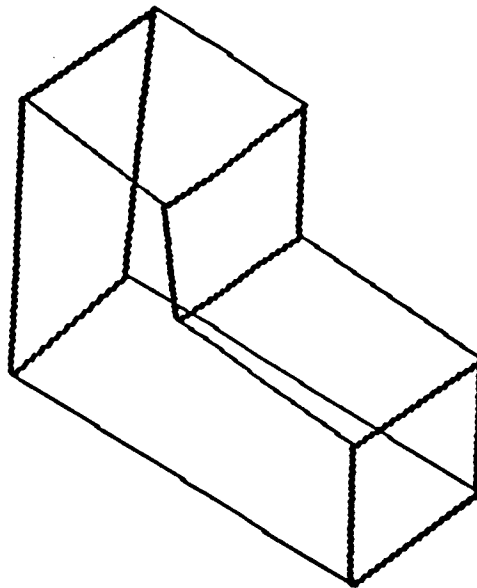


Figure 8: Wireframe Representation of the Test Model

4 CONCLUSION AND FUTURE WORK

A scheme to automatically determine the three dimensional geometry of an object by use a vision system, satisfying certain constraints is presented, developed and tested. An "L" shaped model is chosen as a test object. Experimental results demonstrated the reconstruction of this object geometry within 5 mm discrepancy. The accuracy of the result showed that this technique is quite convenient, efficient to use and can be applied to a wide range of problems in the real world.

Future work would extend this technique to objects with curved surfaces.

ACKNOWLEDGEMENTS

We would like to express our appreciation to Dr. E. Danis and Professor S. H. Shao for their helpful discussion. The work is supported by the Division of Engineering Science in Mech., Structure, and Material Engineering, National Science Foundation, Grant No. MSM 86 15187 "Adaptation of FMS concepts to Construction".

References

- [1] Dana H. Ballard and Christopher M. Brown, *Computer Vision*, Prentice-Hall, Inc., 1982.
- [2] Selspot II, User's Manual Hardware and Software System.
- [3] S. H. Shao, J. C. S. Yang, V. Pavlin, *Corner Detection Using Fussy Sets*, Conference, Santa Barbra, CA, May 1988.
- [4] C. E. Springer, *Geometry and Analysis of Projective Spaces*, W. J. Freeman Co., San Francisco, 1964.

- [5] Irwin Sobel, "*On Calibrating Computer Controlled Cameras for Perceiving 3-D Scenes*", *Artificial Intelligence* 5, pp. 185-198, 1974.

AN INTERACTIVE TESTBED FOR DEVELOPMENT OF EXPERT TOOLS FOR PATTERN RECOGNITION

Stephen W. Wharton
NASA Goddard Space Flight Center
Code 623, Greenbelt, MD, 20769 USA

ABSTRACT

This paper describes the initial implementation of an interactive testbed for development of expert system applications in image processing, i.e., a "toolbox" of procedures designed to facilitate the capture of expert knowledge for region grouping and analysis. The user can elect to interactively enter commands (via a command interpreter) for region manipulation to, in effect, simulate the actions of a hypothetical expert system. The user can then incorporate any rules and procedures as derived from interactive experimentation into customized region processing procedures using the library of utility functions. An iterative technique based on image pyramids is used to compute the initial region segmentation without the use of process parameters. These regions can then be interactively examined and manipulated using the command interpreter.

INTRODUCTION

The process of region extraction and pattern recognition via conventional image processing techniques requires substantial human interaction for enumeration of spatial, spectral, or temporal features for discrimination and iterative refinement of the parameters for recognizing the targets of interest (e.g., edge thresholds, number of clusters, training statistics, etc.). Knowledge-based expert systems offer a means of ultimately reducing the level of human interaction required to achieve accurate results once the appropriate procedural and spectral knowledge bases have been developed. However, the definition and organization of expert knowledge for spectral target recognition is complicated by two factors: 1) it is difficult to focus attention on the discovery of the underlying rules and goals because considerable effort must be devoted towards management of logistics of processing; and 2) the use of processing parameters represents an indirect and ineffective use of knowledge. An alternate method for manipulation of regions is needed that allows users to focus attention on "what to do" to achieve analysis objectives without necessarily having to specify how to do it and that provides the capability for manipulation of regions by inspection rather than by selection of image processing parameters.

This paper describes the Procedure for Interactive Pyramid Segmentation (PIPS) that was developed to assist in the development of knowledge-based spectral target recognition systems by providing an interactive testbed for the identification and quantification of the pertinent procedural and spectral knowledge for the extraction, manipulation, and analysis of regions. The objective

was to produce an initial segmentation (without the customary processing parameters) whose constituent regions could subsequently be examined and manipulated via a library of basic region processing functions. This "toolbox" of functions can be accessed interactively via a command interpreter or called from user-defined region analysis procedures. The intent is that the utility routines could be used to implement customized procedures for automated processing, once an appropriate set of rules has been identified from such interactive experimentation with the command interpreter. The PIPS procedure is written in C and implemented on a Sun workstation. The following sections describe the segmentation technique, summarize the available commands for interactive region manipulation and present an example application. Plans for future development are noted as part of the conclusions.

SEGMENTATION

Background

The functional objective of the segmentation step is to locate regions consisting of one or more pixels that are spatially connected and relatively homogeneous in terms of their average spectral response. An iterative pyramid linking scheme is used to produce a hierarchy of regions without the use of process parameters. A pyramid is an image data structure consisting of multiple levels numbered from 0 to n . The regions at each level in the pyramid represent a property of the image computed at a decreasing spatial resolution. In this application of pyramids, the image property recorded for each region is the mean spectral response. The regions at the bottom level (level 0) represent the image at full spatial resolution, i.e., these regions correspond to single pixels. The regions at each succeeding level (i) are computed as a weighted average of a window of regions at level ($i-1$). For example, windows based on non-overlapping two-by-two kernels yield a pyramid whose spatial resolution decreases by a factor of two between successive levels. The windows are typically overlapped by 50 percent vertically and horizontally so that each region contributes to the average of four "fathers" at the next level, and has 16 "sons" in the level below that contribute to its mean. There is a single region at the top of the pyramid. The number of regions in the pyramid is equal to four-thirds the number of pixels in the original image. The number of pyramid levels is equal to the log (base 2) of the maximum of the number of lines and samples in the image. For additional detail regarding pyramids see Burt et al. (1981) and Tanimoto and Pavlidis (1975).

Approach

The pyramid segmentation and interactive region analysis functions operate on symbolic descriptions of the properties of the overall pyramid and the individual regions within the pyramid as represented by two structures. The overall pyramid processing parameters are recorded in the `image_process_status` (IPS) structure and the region parameters are recorded in the `region_stats` (RS) structure (these structures are listed in Table 1). The original spectral image is only used for

initialization of the IPS and RS parameters. The IPS parameters (**nband**, **nlevel**, **nregion**, **bregion[]**¹, **enregion[]**, **nline[]**, **nsamp[]**, **image_name[]**, **IPS_name[]**, and **RS_name[]**) define various static properties of the pyramid that remain fixed once they have been initialized. The IPS parameters (**atregion**, **page_length**, **page_width**, **class_symbols[]**, **map_symbols[]**, and **lists[][]**) represent variable properties that can be accessed via the command interpreter (CI) for use in interactive region manipulation. The commands available under the CI are described in a subsequent section.

The RS parameters (**id**, **line**, **sample**, and **level**) define various static properties of the region, e.g., unique identifier and location, that remain fixed once they have been initialized. Each region can be connected to at most one parent region at the next higher pyramid level and may have one or more regions that connect to it from the next lower pyramid level. The single region at the top level of the pyramid (with **RS->id = IPS->nregion**)² has no parent and the regions at the base of the pyramid (level 0) have no sons. The number of regions at the bottom level is equal to the number of pixels in the original spectral image from which the pyramid is derived.

The remaining region parameters vary as a function of the hierarchical distribution of son-parent links. The variable region parameters include: **mean[]** - the average spectral response of the image pixels represented by the region; **npix** - the number of pixels represented by the region; **sum[]** - the sum of spectral vectors represented by the region; **parent** - the id of the region at the next level to which the region has been linked, e.g., the bottom-up representation of the region hierarchy; **sons** - the list of regions at the previous pyramid level that have this region as their parent, e.g., the top-down representation of the region hierarchy; **neighbors** - the list of regions that have pixels adjacent to the pixels in this region; **map** - the index of mapping symbol associated with the region (used for producing characters maps); and **class** - the classification index assigned to a region. The properties of the parent regions can be computed in a bottom-up fashion (starting from level 0), from analysis of the regions that are connected to it, e.g., the level (i) region properties can be derived from examination of the level (i-1) regions. An example of a tabular summary of the region_stats parameters is given in Figure 1.

The first step in the segmentation is to compute the initial IPS and RS parameters using the function **PSinit**. The utility functions referenced by **PSinit** and elsewhere are summarized in Table 2. The **PSinit** function is shown in Table 3. The region stats are initialized for each line and sample position within a level, starting with level 0. The level 0 means are copied from the original image and the level 0 neighbors are derived from examination of the regions within a 3-by-3 window surrounding the line, sample coordinates. For all regions, the initial parent region is

¹ Brackets "[]" are used to denote vector variables.

² The notation "->" is used to denote a specific element of a structure, e.g., **RS->id**, refers to the id parameter for a given region.

computed as a function of its grid location at the next higher pyramid level, i.e., by dividing the current region's line and sample coordinates by two. The region is then added to its parent. The region means above level 0 are computed by dividing their sums by the number of pixels. After initialization is complete, the regions represent a square grid of pixels, (e.g., 1 by 1 at level 0, 2 by 2 at level 1, 4 by 4 at level 2, 8 by 8 at level 3 etc.) in which all regions above level 0 nominally have 4 sons. Figure 2 shows the region maps for the first four pyramid levels for a pyramid as initialized by **PSinit**.

The pyramid segmentation scheme is based on the computation of a parent-son relationship between regions in adjacent levels, i.e., each region is linked to a single parent at the next higher pyramid level and has one or more sons linked to its from the next lower pyramid level. The parent-son links define the image segments. For each pixel in the image, the corresponding region can be located at any level of the pyramid by following the parent links until the desired level is reached. Character maps can be produced to show the spatial distribution of these regions and the hierarchy of regions for a given image can be displayed by producing a region map for all pyramid levels. Because the level 0 regions represent single pixels and the top region represents all pixels, only the intermediate pyramid levels are likely to contain any spatial patterns of interest.

The overall pyramid linking segmentation approach is an iterative process in which regions are assigned to fathers having the minimum euclidian distance of separation, region parameters are updated to reflect any changes in the connectivity of their sons, and the process continues until convergence, i.e., no regions are connected to new parents. The **PSmdp** function to perform the iterative region linking is shown in Table 4. The pyramid linking technique was designed to improve the segmentation of linear objects by allowing adjacent spectrally similar regions to coalesce (by connecting them to the same parent) without necessarily being averaged with their backgrounds. Unlike conventional pyramid linking schemes that use a fixed list of potential parent for each region (typically a maximum of four), the list of potential parents is variable, depending on the current set of neighbors and their parents links. Figures 3 and 4 show the level 5 region maps for the variable potential parents and fixed potential parent region linking methods. It would appear from examination of these figures that the variable potential parent method provides improved segmentation of linear features.

COMMAND INTERPRETER

The analyst can interactively manipulate the pyramid regions via the processing functions available under the command interpreter (CI). The processing functions currently available include the following: display or print character maps to examine spatial distribution of regions; produce tabular summaries; split a region from its father; add region to father; and compute average distance to neighbors and sons. The CI allows the user to control the sequence of analysis and the

frequency of interaction by processing single regions, lists of regions, or ranges of regions. Global variables are used to control the display of debug and status information, determine the spatial coordinates of areas to be mapped, and to record the current region of lists of regions of interest. A description of the commands for manipulation of the global variables and the region manipulation commands are given in the following sections.

Global Variable Manipulation

A number of global variables are available (as part of the IPS structure) for the analyst to customize the pyramid processing environment (i.e., direct the display of debug and status information and control map size) and for symbolic manipulation of lists of regions (e.g., compute the fathers, neighbors, or sons of one or more regions). These variables are recorded as part of the image processing status file that is associated with each pyramid to be processed and are saved at the end of an interactive processing session. The commands for manipulation of these variables are summarized below:

atregion:

The variable "atregion" is used to denote the id of the current region of interest. The intent is to avoid the aggravation of having to repetitively enter 6 or 7 digits to examine a particular region. The "@" sub-command is used to set the value of atregion.

e.g., **@ 4555** - sets atregion to 4555
 @ 20 40 - sets atregion to the region at line 20, sample 40, level 0
 @ 10 50 3 - sets atregion to the region at line 10, sample 50, level 3

page_length, page_width:

The variables "page_length" and "page_width" are used to control the dimensions of the window used to display character maps. This allows the user to tailor the maps to the size of the screen being used. The defaults are 60 for page_length and 132 for page width.

e.g., **pl 60** - sets page length to 60 characters
 pw 120 - sets page width to 120 characters

subset:

The "subset" variable is used to define the spatial coordinates (lines and samples) of the area to be mapped.

e.g., **sub 1 40 25 60** - sets area to rectangle defined by begline=1,begsamp=40,
 nline=25, nsamp=60.
 sub 60 90 25 - sets area to 50 pixel square centered at line=60, samp=90.
 sub - subset with no arguments sets area to entire image

A-Z:

Twenty-six vector variables (each having up to 256 entries) are provided for manipulation of lists of regions. The list command syntax is:

{dest_list} {operand} {modifier} {source_list}

dest_list: **"A-Z"** - destination list to be created/modified by this command

operand: **"="** - set dest_list equal to result of applying modifier to source_list
 "+=" - add result of applying modifier to source_list to dest_list
 "-=" - delete result of applying modifier to source_list from dest_list

modifier: **ancestors** - return hierarchy of fathers for each region in list
ascending - sort list in ascending order
descending - sort list in descending order
father - return father for each region in list
neighbors - return neighbors for each region in list
sons - return sons for each neighbor in list
unique - return list with duplicate entries removed

source_list: #, #, ...# - list of region id's
 @ sub-command
 valid list name, e.g., "A-Z"

e.g., **A = 1 4 7 100** - stores four entries in list A
A += 200 - adds region 200 to list
A -= 4 - deletes region 4 from list
B = sons A - stores sons of list A regions in list B
C = neighbors A - stores neighbors of list A regions in list C
C = unique C - removes duplicate entries from list C
D = @ 4 6 7 - stores the region at line 4, sample 6, level 7 in list D

Region Manipulation

The region manipulation commands provide the capability for direct interactive region processing without the use of process parameters. Using these commands the analyst can elect to: classify regions, compute region parameters (e.g., average distance to neighbors and average distance to sons), alter the son-father links, e.g., split sons from or add sons to fathers, produce spatial summary of region distribution (e.g., display or print character maps for specified areas), and produce tabular summary of region statistics. These commands can be applied to single regions, lists of regions, and ranges of regions. The region manipulation commands are summarized below. An example session for region mapping is shown in Figure 5.

add son:

This command adds a new son region to the designated father. The command syntax is {new father region id#} {plus sign} {region id#}.

e.g., **"666 + 32"** - directs the CI to add son region 32 to father region 666.

classify:

This command is used to classify a designated set of regions. The CI uses a pointer to a C function (defined in the main program that calls the CI) to perform the classification.

e.g., **"class *"** - to classify all regions
"class 34 - 78" - to classify regions 34 to 78
"class D" - to classify the regions in list D
"class 343" - to classify region 342
"class" - to classify region given by global variable **atregion**

delete son:

This command deletes a son from its current father. The command syntax is: {minus sign} {region id}.

e.g., **"-220"** - directs the CI to disconnect region 220 from its current father.

distance:

Compute euclidian distance between a region and a list of regions.

e.g., **"dst 100 H"** - compute distances between region 100 and regions in list H
"dst H" - compute distances between region given by global variable **atregion** and regions in list H

exit or quit:

This command used to exit the CI.

e.g., **"ex"** or **"quit"** - to return to routine that called CI

produce region map:

The purpose of this command is to produce a character map of the regions for the image coordinates given by the global variable **subset**. If necessary, the map is subsampled, to fit within the page size given by **page_length** and **page_width**. The command syntax is: {map or pmap} {pyramid_level} {list} {class}

map - display map at the terminal

pmap - print map

pyramid_level - the level of the pyramid at which the regions are mapped

list - "A-Z" or @

class - option to use classification symbol of regions

e.g., **"map C"** - produce a map of the regions in list "C"
"map 4" - produce a map of the regions at pyramid level 4
"map A 2" - produce a map of the regions in list "A" at level 2
"map 0 class" - produce a map using the region level 0 classification symbols

save:

This command is used to manually save current region parameters and image processing parameters.

e.g., **"sa"** - for manual save

show:

This command is used to display the current image processing status.

e.g., **"sh"** - to show status

tabular summary:

This command is used to produce a table of the various region parameters (e.g., mean, mapping symbols, number of pixels etc.) and lists (e.g., neighbors, sons, and father hierarchy) for a given region. An annotated example of the summary table is given in Figure 1.

e.g., **"345"** - produce summary for region 345.
"@@" - produce a summary for the region defined by the IPS->atregion

CONCLUSIONS

The PIPS program was designed to identify regions of spatially connected and spectrally homogeneous pixels and to allow these regions to be interactively manipulated without the use of process parameters. Although it performs a function similar to that of cluster analysis in locating groups of pixels having similar spectral responses, the PIPS procedure has four important differences: 1) contextual information, i.e., the local distribution of spectral values, is taken into

account in the formation of regions; 2) the image is segmented into a hierarchy of regions ranging in size from single pixels to the entire image to facilitate subsequent splitting and merging; 3) the need for indirect control of processing (e.g., specification of the number of regions desired or split and merge thresholds) is avoided by allowing the analyst to edit and label the regions directly via interactive command entry; and 4) support routines are available for the implementation of customized procedures for the application and testing of experimental rules for the processing and classification of regions.

The current version of PIPS was developed for use with multispectral image data that satisfy two assumptions: 1) regions of interest can be discriminated from one another according to their average spectral response; and 2) the regions are relatively large compared to the pixel size. At present, PIPS is not designed to discriminate regions on the basis of second-order spectral statistics such as variance or texture, nor to analyze fractions of cover types occurring within mixed pixels. Areas for future development of PIPS are summarized below:

- o Integrate color image display for viewing false color images with color graphics overlay of the region boundaries. Employ cursor pointing device for region selection.
- o Develop utility function to form arbitrary groups of regions without having to connect them through the same parent in the region hierarchy.
- o Incorporate additional similarity measures for pyramid linking, e.g., group regions according to local contrast, shape, spectral variation, and texture. Also develop method to group regions that are not necessarily adjacent to one another.
- o Develop capability to analytically determine the probable combinations and proportions of the spectral constituents that comprise spectrally mixed regions.

REFERENCES

- Burt, P. J., T. Hong, and A. Rosenfeld, "Segmentation and Estimation of Image Region Properties Through Cooperative Hierarchical Computation," IEEE Trans. on Systems, Man, and Cybernetics, Vol. 11, No. 12, 1981, pp. 802-809.
- Tanimoto, S. L. and T. Pavlidas, "A Hierarchical Data Structure for Picture Processing," Computer Graphics and Image Processing, Vol. 4, 1975, pp. 104-119.

Table 1: Description of the image_process_status (IPS) and region_stats (RS) structures used in pyramid segmentation.

```

struct image_process_status
{
    int atregion;           /* id of current region of interest */
    int nband, nlevel, nregion; /* number of bands, level, and regions in pyramid */
    int page_length, page_width; /* max page length and width (characters) for image maps */

    char class_symbols[];   /* classification map symbols */
    char mapping_symbols[]; /* region mapping symbols */
    int bregion[], enregion[]; /* index of first and last region per pyramid level */
    int nline[], nsamp[];   /* number of lines and samples represented per level */
    int subset[];          /* coordinates of area to be mapped */
    int lists[][];          /* lists of integers referenced by 'A-Z' via CI */
    char image_name[], IPS_name[], RS_name[]; /* associated image, IPS, and RS file names */
};

struct region_stats
{
    int id;                 /* region identification number */
    int mdst_parent;        /* minimum distance parent */
    int parent;             /* parent of region */
    int npix;               /* number of pixels represented by region */
    int line, sample, level; /* region coordinates */
    int nneig, nson;        /* number of neighbors and sons */
    int class, map;         /* classification and map symbol index */
    int neighbors[], sons[]; /* lists of neighboring and son regions */
    int mean[], sum[];      /* mean and sums of pixel spectral vectors represented by region */
};

```

Table 2: List of utility functions for region processing that are used by the routines PSinit and PSmdp (Tables 3 and 4). The function parameters are given in brackets "{}".

addson:	add son region to parent region's son list, neighbor list, npix, and sum; {IPS, parent, son}.
comp_mean:	compute mean by dividing RS->mean by RS->npix; {IPS, region}.
comp_mdr:	compute the minimum distance region, i.e., the member of the region list having the minimum euclidian distance to the region denoted by RS; {IPS, region_list, RS}.
delson:	delete son region from parent region's son list, neighbor list, npix, and sum; {IPS, parent, son}.
distance:	compute euclidian distance between means of regions i and j; {IPS, i, j}.
get_IPS:	read image_process status for a given image; {image_name}.
get_pixel:	read spectral response vector for pixel at given coordinates; {IPS, line, sample}.
get_RS:	read region_statistics structure from disk for designated region; {IPS, region}.
grid_neighbors:	compute list of neighbor regions from the 3-by-3 window of pixels around the coordinates (RS->line, RS->sample); {IPS, RS}.
init_RS:	initialize npix and sum vector to zero; {IPS, RS}.
map_index:	compute mapping symbol index so that no two neighboring regions have the same index; {RS}.
parents:	form parent list by recording parent for each member of the region list; {IPS, region_list, parent_list}.
PSinit:	initialize pyramid for segmentation (listed in Table 3); {image_name, nband, nline, nsamp}.
PSmdp:	compute pyramid segmentation by iterated linking of regions to minimum distance parents (listed in Table 4); {image_name, nband, nline, nsamp}.
put_IPS:	write image_process status; {IPS}.
put_RS:	write region_statistics structure on disk; {IPS, RS}.

Table 3: Listing for function PSinit that, given the name and the number of bands, lines, and samples in the original image, computes the initial region_statistics parameters for each region and the image_process_status parameters for the overall pyramid.

```

PSinit(image_name,nband,nline,nsamp)
{
    atlevel = atregion = nregion = 0
    IPS = get_IPS(image_name)
    do
    {
        IPS->bregion[atlevel] = nregion + 1
        nregion += (nline * nsamp)
        IPS->enregion[atlevel] = nregion
        IPS->nline[atlevel] = nline; IPS->nsamp[atlevel] = nsamp
        below_top = ((nline > 1) && (nsamp > 1))
        nline = (nline + 1) / 2; nsamp = (nsamp + 1) / 2

        for (atline = 1; atline <= IPS->nline[atlevel]; atline++)
        {
            for (atsamp = 1; atsamp <= IPS->nsamp[atlevel]; atsamp++)
            {
                atregion++
                RS = get_RS(IPS,atregion)
                RS->id = atregion
                RS->line = atline; RS->sample = atsamp; RS->level = atlevel
                RS->npix = (atlevel == 0) ? 1 : 0

                /* compute index of parent region at next pyramid level */
                parent_line = (atline + 1) / 2; parent_samp = (atsamp + 1) / 2
                parent = nregion + (nline * (parent_line - 1)) + parent_samp
                RS->parent = (below_top) ? parent : 0

                if (RS->level == 0)
                {
                    /* copy mean and sum from pixel */
                    mean = get_pixel(IPS,RS->line,RS->sample)
                    for (atband = 0; atband < IPS->nband; atband++)
                        RS->mean[atband] = RS->sum[atband] = mean[atband]

                    grid_neighbors(RS) /* compute list of neighbors */
                }
                else
                    comp_mean(IPS,RS) /* compute mean from accumulated sum */

                RS->map = map_index(RS) /* compute region mapping symbol */
                if (RS->parent) addson(IPS,RS->parent,atregion) /* add region to parent */
                put_RS(IPS,RS) /* record region stats */
            }
        }
        atlevel++
    }
    while (below_top)

    IPS->nband = nband; IPS->nlevel = atlevel; IPS->nregion = nregion
    put_IPS(IPS)
}

```

Table 4: Instructions for procedure PSmdp to perform the pyramid segmentation by iteratively linking each region to its minimum distance parent.

```

PSmdp(IPS)
{
    beg_region = 1; end_region = IPS->nregion - 1
    beg_parent = IPS->bregion[1]; end_parent = IPS->nregion

    do
    {
        nsons_modified = 0

        /* initialize nneig, npix, nson and sum of parent regions */
        for (parent = beg_parent; parent <= end_parent; parent++)
        {
            RS = get_RS(IPS,parent)
            init_RS(IPS,RS)
            put_RS(IPS,RS)
        }

        /* assign regions to minimum distance parent */
        for (region = beg_region; region <= end_region; region++)
        {
            RS = get_RS(IPS,region)

            /* derive list of candidate parents by recording current parent for each neighbor */
            parents(IPS,RS->neighbors,parents)
            RS->mdst_parent = comp_mdr(IPS,parents,RS)    /* compute minimum distance parent */

            addson(IPS,RS->mdst_parent,region)           /* add region to parent */
            if (RS->mdst_parent != RS->parent) nsons_modified++
        }

        /* update links to parents */
        for (region = beg_region; region <= end_region; region++)
        {
            RS = get_RS(IPS,region)
            RS->parent = RS->mdst_parent
        }

        /* update parent means */
        for (parent = beg_parent; parent <= end_parent; parent++)
        {
            RS = get_RS(IPS,region)
            comp_mean(IPS,RS)
            put_RS(IPS,RS)
        }
    }
    while (nsons_modified > 0)
}

```


[illegible]

	36	39	42	45	48	51	54	57	60	63	66	69	72	75	78	81	84	87	90	93
	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
28	+	j	j	j	j	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h
29	+	h	h	j	j	<	h	h	h	h	h	h	h	h	h	h	h	h	h	h
30	+	h	h	h	j	j	<	<	h	h	h	h	h	h	h	h	h	h	h	h
31	+	h	h	h	h	<	j	h	h	h	h	h	h	h	h	h	h	h	h	h
32	+	h	h	h	h	h	<	j	h	h	h	h	h	h	h	h	h	h	h	h
33	+	h	h	h	h	h	<	j	j	j	j	j	j	j	j	j	j	j	j	j
34	+	h	h	h	h	h	j	j	j	j	j	j	j	j	j	j	j	j	j	j
35	+	h	h	h	h	h	j	<	j	j	j	j	j	j	j	j	j	j	j	j
36	+	h	h	<	h	h	h	j	h	h	h	h	h	h	h	h	h	h	h	h
37	+	h	h	<	h	h	h	j	h	h	h	h	h	h	h	h	h	h	h	h
38	+	h	h	h	h	h	j	j	j	j	j	j	j	j	j	j	j	j	j	j
39	+	h	h	h	h	h	j	*	j	j	j	j	j	j	j	j	j	j	j	j
40	+	h	h	h	h	j	*	j	j	j	j	j	j	j	j	j	j	j	j	j
41	+	h	h	h	<	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
42	+	<	<	<	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
43	+	h	j	<	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
44	+	h	h	j	<	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
45	+	j	j	j	j	*	j	j	j	j	j	j	j	j	j	j	j	j	j	j
46	+	j	j	j	j	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<
47	+	h	h	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<
	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
	36	39	42	45	48	51	54	57	60	63	66	69	72	75	78	81	84	87	90	93

Figure 4: Character maps for pyramid level (5) regions for five meter resolution, eight band Thematic Mapper Simulator data covering buildings 19 and 20 at the NASA Goddard Space Flight Center in Greenbelt, Maryland. The segmentation shown in the top image was generated using the region linking with a variable list of potential parents. The segmentation shown in the bottom image was generated using the region linking with a fixed list of potential parents.

```

--->sub 28 36 13 30      (define mapping subset as upper left of image in Figure 4)
..ci-sub:  sub 1(28,47), s(36,65)
--->A = @ 34 54 5        (define list A as rectangular region of j's)
--->m A                  (map region in list A)

```

```

      36 39 42 45 48 51 54 57 60 63
      +---+---+---+---+---+---+---+---+---+---+
28 +                                     + 28
29 +                                     + 29
30 +                                     + 30
31 +                                     + 31
32 +                                     + 32
33 +                                     + 33
34 +                                     + 34
35 +                                     + 35
36 +                                     + 36
37 +                                     + 37
38 +                                     + 38
39 +                                     + 39
40 +                                     + 40
      +---+---+---+---+---+---+---+---+---+---+
      36 39 42 45 48 51 54 57 60 63

```

```

--->B = s A                (store sons of A in B)
--->C = s B                (store sons of B in C)
--->D = s C                (store sons of C in D and map D)
--->m D

```

```

      36 39 42 45 48 51 54 57 60 63
      +---+---+---+---+---+---+---+---+---+---+
28 +                                     + 28
29 +                                     + 29
30 +                                     + 30
31 +                                     + 31
32 +                                     + 32
33 +                                     + 33
34 +                                     + 34
35 +                                     + 35
36 +                                     + 36
37 +                                     + 37
38 +                                     + 38
39 +                                     + 39
40 +                                     + 40
      +---+---+---+---+---+---+---+---+---+---+
      36 39 42 45 48 51 54 57 60 63

```

Figure 5: Annotated example session log where "--->" denotes the prompt from the command interpreter and comments are given in parenthesis. The command sequence was entered to show the descendents of the large rectangular region in the center of the subset (NASA GSFC building #20).

PARALLEL AND DISTRIBUTED COMPUTATION FOR FAULT-TOLERANT OBJECT RECOGNITION

by

Harry Wechsler
Department of Computer Science
George Mason University
Fairfax, VA 22030

Abstract: We suggest the distributed associative memory (DAM) model for distributed and fault-tolerant computation as it relates to object recognition tasks. The fault-tolerance is with respect to geometrical distortions (scale and rotation), noisy inputs, occlusion/overlap, and memory faults. We have developed an experimental system for fault-tolerant structure recognition which shows the feasibility of such an approach. We further extended our approach to the problem of multisensory data integration and applied it successfully to the recognition of colored polyhedral objects.

1. Introduction

The challenge of the visual recognition problem stems from the fact that the projection of an object onto an image can be confounded by several dimensions of variability such as uncertain perspective, changing orientation and scale, sensor noise and occlusion, and non-uniform illumination. A vision system must not only be able to sense the identity of an object despite this variability, but must also be able to characterize such variability—because the variability inherently carries much of the valuable information about the world.

Our goal is to derive the functional characteristics of image representations suitable for invariant recognition using distributed processing methods. One has to seek appropriate transformations such that interactions between the internal structure of the resulting representations yield invariant recognition. As Simon (1982) points out, all mathematical derivation can be viewed simple as a change of representation, making evident what was previously true but obscure. Solving a problem then means transforming it so as to make the solution transparent.

Intelligent behavior can be considered as an information processing task which must be understood at three levels (Marr, 1982). First, the basic computational theory specifies what is the task, why is it appropriate, and what is the strategy by which it can be carried out. Second, the representation and algorithm specify how the computational theory can be implemented in terms of input, output, and transformations. Third, the hardware specifies the actual implementation. It is clear that the task determines the mixture of representations and algorithms. It is also clear that a good fit between the three levels is highly desirable and beneficial. The representation and algorithms, to be discussed in detail later on, are characteristic of an emerging AI trend, that of parallel and distributed computation, known as neural networks (NN) or artificial neural systems (ANS) (Hecht-Nielsen, 1986).

Much of the work in computer vision has been centered on the use of single sensory cues such as shading or luminance gradients, for interpretation of scene parameters. It is clear that multiple cues could be used to enhance visual recognition. Our goal is to form consistent scene descriptions by combining invariant representations from multiple sensors.

The recognition process should match a derived input representation of the short term memory (STM) against long term memory (LTM). We suggest the LTM organization in terms of distributed associative memory (DAM). The DAM is related to generalized match filters (GMF) (Caulfield and Weinberg, 1982) and synthetic discriminant functions (SDF) (Hester and Casasent, 1981). Like them, DAMs attempt to capture a distributed representation which averages over the variations belonging to the same class. DAMs allow for the implicit representation of structural relationships and contextual information, helpful constraints for choosing among different interpretations. Finally, because information is distributed in the memory, the overall function of the memory becomes resistant to noise, faults in memory, and degraded stimulus key vectors.

This paper is concerned with combining multiple sources of information using invariant transformations. The first section is an overview of relevant research concerning data fusion and invariant object recognition from the areas of biological perception, computational vision and neural networks. The second outlines our proposed recognition system. The third section demonstrates the feasibility of our method of data fusion by presenting some experimental results on real images. We conclude by attempting to address the 3-D recognition problem.

2. Background

Our review begins by examining biological vision systems for evidence of the modularity of image analysis and the recombination of the modules. This examination is continued by presenting some recent models of data integration found in the computer vision literature. This section concludes with an outline of neural networks which we think will be a force for bridging the gap between biological and computer vision systems.

2.1. Biological Vision

Psychophysical and physiological experiments suggest biological vision is modular in design. Experiments on human visual perception using random-dot stereograms (Julesz, 1975) indicate that humans have the ability to interpret images in 3-dimensions using only cues such as stereopsis and texture. Psychophysical tests of human infants also suggest a modularity of sensitivity to different cues. According to Yonas, Arterberry, and Granrud (1987) the development of an infant follows a definite pattern of sensitivity. Retinal size and motion parallax are early cues used by infants to discriminate depth. These are followed by binocular parallax and then pictorial depth. Staggered development of the ability to use different depth cues indicated different processing is occurring in parallel on the same visual stimulus. Recent work in neurosciences, reviewed by Van Essen and Maunsell (1983), show there are a large number of well defined subdivisions in the visual cortex. Evidence suggests the existence of at least two major functional streams, one related to the analysis of motion and the other related to the analysis of form and color. These functional streams are independent in many respects and the processing of information is being done continuously and in parallel. The question that immediately comes to mind is if these processes are independent and modular how are they integrated to maintain a continuous, complete, coherent perception of the world.

Biological perception of space is not limited to cues within the single modality of vision but can also be driven by entirely different modalities. An example of the influence of extravisual sources in spatial orientation tasks can be found in the barn owl. In barn owls the auditory system plays a crucial role in prey capture; since it hunts at night visual cues are of little value. As a consequence owls rely heavily on their sense of hearing to localize prey and guide their attack. The owl have cells in its optic tectum (the main visual center) which have topographic representations of visual space that are bimodally sensitive to auditory and visual stimulation (Knudsen, 1982). The topographic representations of visual space depend on point to point projections from the retina. The map of auditory space is an emergent property of higher order processing. The auditory system must derive its map from the relative patterns of auditory input arriving at each ear (Knudsen and Konishi, 1978). Despite different ways of deriving the spatial information, the visual and auditory map were found to be remarkably similar and closely aligned. Other studies of bimodally sensitive cells have found the interactions between the modalities can be non-linear and complex (Hartline et. al., 1978). How can information from different modalities be combined to extract appropriate parameters from the environment?

2.2. Computer Vision

A problem immediately related to recognition is that extracting relevant information about the 3-D environment from 2-dimensional projections is inherently underconstrained. Two basic mathematical approaches have been suggested. Poggio (1985) suggests regularization to solve the ill-posed problems presented by early vision. Regularization uses a-priori knowledge in the form of variational principles or statistical properties of the solution space to enforce constraints derived from physical analysis. Another method, that of CAD for vision will approach the same problem by solid modeling in terms of only edges-based primitives, disregarding any surface information. Such an attempt still has to define what the primitives are. Recent work by Biederman (1987) suggests that there is a relatively small repertoire of basic primitives ('geons') needed to build-up 3-D shapes.

Computer vision systems have been less than successful in finding approaches for integration of spatial information from multiple sources. Most of the research in the past ten years has centered around using a single image cues. The techniques are fragile and tend to work only on very simple domains because they rely on underlying assumptions about the world which are insufficient or invalid for complex scenes. In essence each separate module imparts a grain of truth but none are entirely reliable alone.

The AI approach toward solving recognition and data fusion type of problems is mainly that of symbolic processing (Garvey and Lowrance, 1983). Uncertainty can be handled through the Dempster-Shafer model of evidential reasoning (Gordon and Shortiffe, 1984) which is a good method for integration of several sources. However, defining the corresponding intervals of confidence is not a trivial task. Furthermore, there is no clear way for such methods to handle the intrinsic/iconic images of low-level vision.

2.3. Neural Networks

Neural networks were developed originally to account for biological memory systems. They implement a type of distributed representation and computation, where a large number of highly interconnected 'simple' processing elements (PE) operate in parallel. NNs provide a good fit among

the three levels at which retrieval and/or recognition tasks should be understood. Through the NNs' collective dynamics, the emergent behavior which is the result of both competition and cooperation between neighboring PEs, yields the optimal solution, subject to contextual constraints implicitly embedded in the net of interconnections. Such an approach is akin to relaxation.

An excellent collection of papers summarizing recent work on Parallel Distributed Processing (PDP) (McClelland and Rumelhart, 1986) includes some reflections made by Norman regarding the major advantages offered by distributed computation. Specifically, learning can be made continuous, natural, and fundamental to such models of computation. New conceptualizations are reflected by qualitatively different state configurations. Information is passed among the units, not by messages, but by activation values, and by scalars instead of symbols. The input (image) interpretation is achieved through the detection of highly active states. From such a discussion, one can appreciate the transition between the traditional AI 'scheme/frame' representational tool to a more complex and dynamic level of knowledge representation. The NN approach fits well with some of the requirements for both recognition and sensor integration. It can handle iconic representations, is modular and allows a transformational approach for capturing the invariants needed for recognition.

3. System Outline

This section describes our invariant object recognition system in detail. We begin by examining the preprocessing system which produces the vectors associated by the distributed associative memory. Then the distributed associative memory and our method of fusing multiple sources of information are described and analyzed. A block diagram of the system is shown in Figure 1.

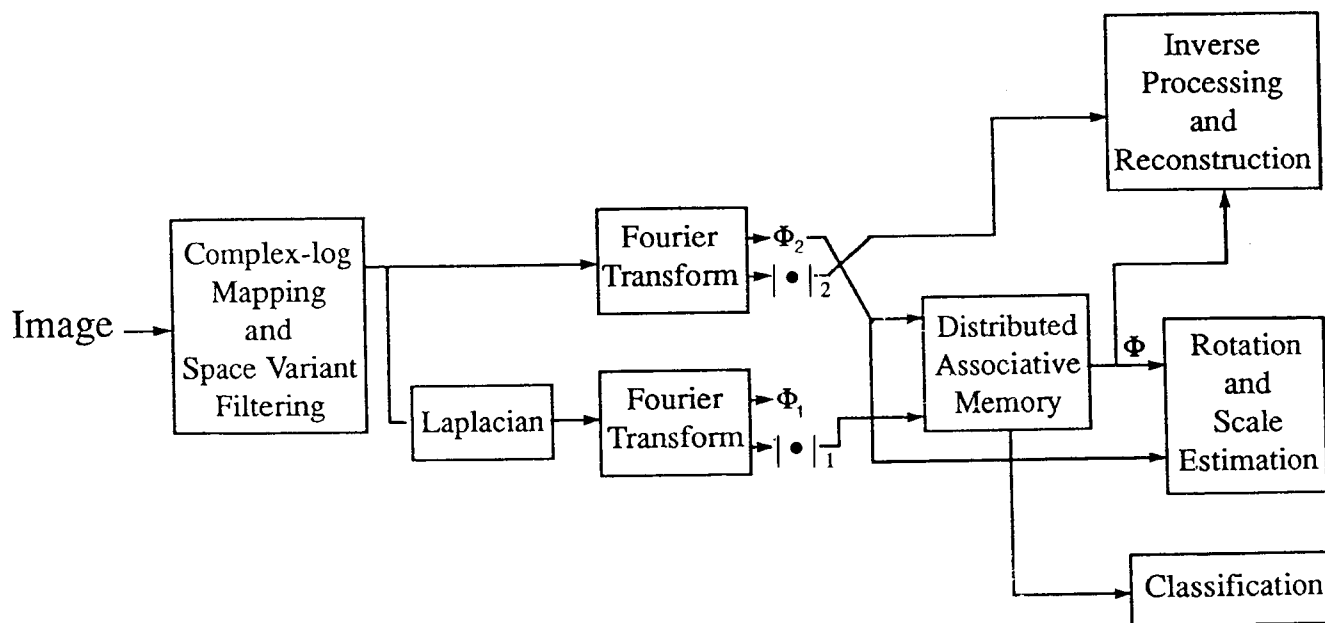


Figure 1. Block Diagram of the System.

3.1. Invariant Representation

The goal of this subsection is to examine the various components used to produce the vectors which are associated in the distributed associative memory. The block diagram which describes the various functional units involved in obtaining an invariant image representation is shown in Figure 1. The image is complex-log conformally mapped so that rotation and scale changes become translation in the transform domain. Along with the conformal mapping, the image is also filtered by a space variant filter to reduce the effects of aliasing. The conformally mapped image is then processed through a Laplacian in order to solve some problems associated with the conformal mapping (See 3.1.3). The Fourier transform of both the conformally mapped image and the Laplacian processed image produce the four output vectors. The magnitude output vector $|\bullet|_1$ is invariant to linear transformations of the object in the input image. The phase output vector ϕ_2 contains information concerning the spatial properties of the object in the input image.

3.1.1. Complex-Log Mapping and Space Variant Filtering

The first box of the block diagram given in Figure 1 consists of two components: complex-log mapping and space variant filtering. Complex-log mapping transforms an image from rectangular coordinates to polar exponential coordinates. This transformation changes rotation and scale into translation. If the image is mapped onto a complex plane then each pixel (x,y) on the Cartesian plane can be described mathematically by $z = x + jy$. The complex-log mapped points w are described by

$$w = \ln(z) = \ln(|z|) + j\theta_z \quad (1)$$

where $|z| = (x^2 + y^2)^{1/2}$ and $\theta_z = \tan^{-1}(y/x)$.

Our system sampled 256×256 pixel images to construct 64×64 complex-log mapped images. Samples were taken along radial lines spaced 5.6 degrees apart. Along each radial line the step size between samples increased by powers of 1.08. These numbers are derived from the number of pixels in the original image and the number of samples in the complex-log mapped image. An excellent examination of the different conditions involved in selecting the appropriate number of samples for a complex-log mapped image is given in (Massone et. al., 1985) The non-linear sampling can be split into two distinct parts along each radial line. Toward the center of the image the samples are dense enough that no anti-aliasing filter is needed. Samples taken at the edge of the image are large and an anti-aliasing filter is necessary. The image filtered in this manner has a circular region around the center which corresponds to an area of highest resolution. The size of this region is a function of a number of angular samples and radial samples. The filtering is done, at the same time as the sampling, by convolving truncated Bessel functions with the image in the space domain. The width of the Bessel functions main lobe is inversely proportional to the eccentricity of the sample point.

A problem associated with the complex-log mapping is sensitivity to center misalignment of the sampled image. Small shifts from the center causes dramatic distortions in the complex-log mapped image. Our system assumes that the object is centered in the image frame. Slight misalignments are considered noise. Large misalignments are considered as translations and could be accounted for by changing the gaze in such a way as to bring the object into the center of the frame (see 4.2). The decision about what to bring into the center of the frame is an active function and should be determined by the task. An example of a system which could be used to guide the translation process was developed by Anderson *et al* (1985). Their pyramid system analyzes the input image at different temporal and spatial resolution levels. Their smart sensor was then able to shift its fixation such that interesting parts of the image (i.e. something large and moving) was brought into the central part of the frame for recognition.

3.1.2. Fourier Transform

The second box in the block diagram of Figure 1 is the Fourier transform. The Fourier transform of a 2-dimensional image $f(x,y)$ is given by

$$F(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) e^{-j(ux+vy)} dx dy \quad (2)$$

and can be described by two 2-dimensional functions corresponding to the magnitude $|F(u,v)|$ and phase $\Phi_F(u,v)$. The magnitude component of the Fourier transform which is invariant to translation, carries much of the contrast information of the image. The phase component of the Fourier transform carries information about how things are placed in an image. Translation of $f(x,y)$ corresponds to the addition of a linear phase component. The complex-log mapping transforms rotation and scale into translation and the magnitude of the Fourier transform is invariant to those translations so that $|\bullet|_1$ will not change significantly with rotation and scale of the object in the image.

3.1.3. Laplacian

The Laplacian that we use is a difference-of-Gaussians (DOG) approximation to the $\nabla^2 G$ function as given by Marr (1982).

$$\nabla^2 G = \frac{1}{\pi\sigma^4} [1 - r^2/2\sigma^2] e^{-r^2/2\sigma^2} \quad (3)$$

The result of convolving the Laplacian with an image can be viewed as a two step process. The image is blurred by a Gaussian kernel of a specified width σ . Then the isotropic second derivative of the blurred image is computed. The width of the Gaussian kernel is chosen such that the conformally mapped image is visible—approximately 2 pixels in our experiments. The Laplacian sharpens the edges of the object in the image and sets any region that did not change much to zero. Below we describe the benefits from using the Laplacian.

The Laplacian eliminates the stretching problem encountered by the complex-log mapping due to changes in object size. When an object is expanded the complex-log mapped image will translate. The pixels vacated by this translation will be filled with more pixels sampled from the center of the scaled object. These new pixels will not be significantly different than the displaced pixels so the result looks like a stretching in the complex-log mapped image. The Laplacian of the complex-log mapped image will set the new pixels to zero because they do not significantly change from their surrounding pixels. The Laplacian eliminates high frequency spreading due to the finite struc-

ture of the discrete Fourier transform and enhances the differences between memorized objects by accentuating edges and de-emphasizing areas of little change.

3.2 Distributed Associative Memory (DAM)

The particular form of distributed associative memory that we deal with in this paper is a memory matrix which modifies the flow of information. Stimulus vectors are associated with response vectors and the result of this association is spread over the entire memory space. Distributing in this manner means that information about a small portion of the association can be found in a large area of the memory. New associations are placed over the older ones and are allowed to interact. This means that the size of the memory matrix stays the same regardless of the number of associations that have been memorized.

The above discussion illuminates several properties of distributed associative memories which are different from the more traditional ones about memory. Because the associations are allowed to interact with each other an implicit representation of structural relationships and contextual information can develop, and as a consequence a very rich level of interactions can be captured. Since there are few restrictions on what vectors can be associated there can exist extensive indexing and cross-referencing in the memory. Since the information is distributed, the overall function of the system is resistant to faults in the memory and degraded stimulus vectors. Distributed associative memory captures a distributed representation which is context dependent. This is quite different from the simplistic behavioral model (Hebb, 1949).

3.2.1. Construction and Recall

The construction stage assumes that there are n pairs of m -dimensional vectors that are to be associated by the distributed associative memory. This can be written as

$$M\bar{s}_i = \bar{r}_i \text{ for } i = 1, \dots, n \quad (4)$$

where \bar{s}_i denotes the i^{th} stimulus vector and \bar{r}_i denotes the i^{th} corresponding response vector. We want to construct a memory matrix M such that when the k^{th} stimulus vector \bar{s}_k is projected onto the space defined by M the resulting projection will be the corresponding response vector \bar{r}_k . More specifically we want to solve the following equation:

$$MS = R \quad (5)$$

where $S = [\bar{s}_1 | \bar{s}_2 | \dots | \bar{s}_n]$ and $R = [\bar{r}_1 | \bar{r}_2 | \dots | \bar{r}_n]$. A unique solution for this equation does not necessarily exist for any arbitrary group of associations that might be chosen. Usually, the number of associations n is smaller than m , the length of the vector to be associated, so the system of equations is underconstrained. The constraint used to solve for a unique matrix M is that of minimizing the square error, $\|MS - R\|^2$, which results in the solution

$$M = RS^+ \quad (6)$$

where S^+ is known as the Moore-Penrose generalized inverse of S (Kohonen, 1984)

The recall operation projects an unknown stimulus vector \bar{s} onto the memory space M . The resulting projection yields the response vector \bar{r}

$$\bar{r} = M\bar{s} \quad (7)$$

If the memorized stimulus vectors are independent and the unknown stimulus vector \bar{s} is one of the memorized vectors \bar{s}_k , then the recalled vector will be the associated response vector \bar{r}_k . If the memorized stimulus vectors are dependent, then the vector recalled by one of the memorized stimulus vectors will contain the associated response vector and some crosstalk from the other stored response vectors. The resulting noise or crosstalk in the output is due to the similarity of the memorized vectors.

The recall can be viewed as the weighted sum of the response vectors. The recall begins by assigning weights according to how well the unknown stimulus vector matches with the memorized stimulus vector using a linear least squares classifier. The response vectors are multiplied by the weights and summed together to build the recalled response vector. The recalled response vector is usually dominated by the memorized response vector that is closest to the unknown stimulus vector. The distributed associative memory will have interactions between the different associations and this allows some generalization of responses to previously unknown stimulus.

Fault tolerance is a byproduct of the distributed nature and error correcting capabilities of the distributed associative memory. By distributing the information, no single memory cell carries a significant portion of the information critical to the overall performance of the memory. Assume that there are n associations in the memory and each of the associated stimulus and response vectors have m elements. This means that the memory matrix has m^2 elements. Also assume that the noise that is added to each element of a memorized stimulus vector is independent, zero mean, with a variance of σ_i^2 . The recall from the memory is then

$$\bar{r} = \bar{r}_k + \bar{v}_o = M(\bar{s}_k + \bar{v}_i) = \bar{r}_k + M\bar{v}_i \quad (8)$$

where \bar{v}_i is the input noise vector and \bar{v}_o is the output noise vector. The ratio of the average output noise variance to the average input noise variance is

$$\sigma_o^2/\sigma_i^2 = \frac{1}{m} \text{Tr}[MM^T] \quad (9)$$

For the autoassociative case this simplifies to

$$\sigma_o^2/\sigma_i^2 = \frac{n}{m} \quad (10)$$

This says that when a noisy version of a memorized input vector is applied to the memory the recall is improved by a factor corresponding to the ratio of the number of memorized vectors to the number of elements in the vectors. For the heteroassociative memory matrix a similar formula holds as long as n is less than m (Stiles and Denz, 1985).

$$\sigma_o^2/\sigma_i^2 = \frac{1}{m} \text{Tr}[RR^T]\text{Tr}[(S^TS)^{-1}] \quad (11)$$

Another way of viewing this error correcting process is to notice that the memory matrix is the orthogonal projection matrix for the set of stimulus vectors. The noise vector in this m -dimensional space will be projected onto the space spanned by the n memorized vectors. The parts of the noise vector that are orthogonal to the n memorized stimulus vectors will be lost and this accounts for the noise reduction in the output recall vector.

3.3 Data Fusion

We suggest next using distributed associative memory to integrate visual information from multiple cues or sources for performing image interpretation tasks. A source amounts to a 2-dimensional function extracted from an image which carries some distinct information about the 3-dimensional scene. In our particular case the sources of information can take on two forms: direct or derived. A direct source is one where the information is directly in registration with the iconic image. Examples would be infra-red images, range images, spectral band images, etcetera. A derived source is one which requires some type of preprocessing before the spatial information is made explicit. Preprocessing examples are stereopsis and optical flow derivation. Integration means combining information from several distinct sources to produce a response which is possibly quite different from the input information. An example of integration would be the ability of human vision to extract reliable viewer-centered depth information from the binocular and motion information contained in a set of images. In this case the 'sensors', stereopsis and optical flow, are distinct but both carry geometric information about the scene. In order for information from distinct sources to be integrated at some point they must speak the same language.

Looking for a general method for combining multiple distinct sources into a unified interpretation is a problem which has been examined from many angles. Statistical approaches such as Garvey and Lowrance's (1983) threat assessment for battle management which uses Dempster/Shaffer model for source integration, need much a-priori information about the statistical variations between the different source measurements and are computationally very expensive. Hybrid approaches such as Waxman and Duncan's (1986) stereomotion, use extensive knowledge about how the sources vary with each other in order to develop their algorithm, and as a result are not general or easily extensible to the addition of other sources. Both of these methods need higher level processing to handle conflicts in source interpretation.

Our approach does not fit entirely into either of these classes. We assume that the output of the sources vary consistently for a given input. Let $S = [\bar{s}_1 | \bar{s}_2 | \dots | \bar{s}_n]$ and $R = [\bar{r}_1 | \bar{r}_2 | \dots | \bar{r}_n]$ be the stimulus and response matrices respectively. Each stimulus vector is made up of elements coming from the different sources $\bar{s}_i^T = [\bar{s}_{i1}^T | \bar{s}_{i2}^T | \dots | \bar{s}_{i\ell}^T]$ where \bar{s}_{ij}^T consists of elements from the j^{th} source. The memory matrix is constructed in the same fashion as before, $M = RS^+$.

Integration of sources in this manner has several positive properties. First, it is quite general and as such it is easily extensible to multiple sources. If new sources need to be added they simply augment the previous stimulus vector structure. Second, prioritizing the influence that a single source can have on the output can be done by adjusting the size and/or the quantization of elements that source donates to the stimulus vectors. The section dedicated to experimental results discusses how statistical analysis can 'prioritize' sources according to both their relevance and reliability.

4. Experiments

In this section we discuss the result of computer simulations of our system. Images of objects are first preprocessed through the subsystem outlined in subsection 3.2. The output of such a subsystem is four vectors: $|\bullet|_1$, Φ_1 , $|\bullet|_2$, and Φ_2 . We construct the memory by associating the stimulus vector $|\bullet|_1$ with the response vector Φ_2 for each object in the database. To perform a recall from the memory the unknown image is preprocessed by the same subsystem to produce the vectors $|\tilde{\bullet}|_1$, $\tilde{\Phi}_1$, $|\tilde{\bullet}|_2$, and $\tilde{\Phi}_2$. The resulting stimulus vector $|\tilde{\bullet}|_1$ is projected onto the memory matrix to produce a response vector which is an estimate of the memorized phase $\hat{\Phi}_2$. The estimated phase vector $\hat{\Phi}_2$ and the magnitude $|\tilde{\bullet}|_1$ are used to reconstruct the memorized object. The difference between the estimated phase $\hat{\Phi}_2$ and the unknown phase $\tilde{\Phi}_2$ is used to estimate the amount of rotation and scale experienced by the object.

4.1. Invariant Recognition

The database of images consists of twelve objects: four keys, four mechanical parts, and four leaves. The objects were chosen for their essentially two-dimensional structure. Each object was photographed using a digitizing video camera against a black background. We emphasize that all of the images used in creating and testing the recognition system were taken at different times using various camera rotations and distances. The images are digitized to 256×256 , eight bit quantized pixels, and each object covers an area of about 40×40 pixels. This small object size relative to the background is necessary due to the non-linear sampling of the complex-log mapping. The objects were centered within the frame by hand. This is the source of much of the noise and could have been done automatically using the object's center of mass or some other criteria determined by the task. The orientation of each memorized object was arbitrarily chosen such that their major axis was vertical. The 2-dimensional images that are the output from the invariant representation subsystem are scanned horizontally to form the vectors for memorization.

The first example of the operation of our system is shown in Figure 2. Figure 2a) is the image of one of the key as it was memorized. Figure 2b) is the unknown object presented to our system. The unknown object in this case is the same key that has been scaled. Figure 2c) is the recalled, reconstructed image. The rounded edges of the recalled image are artifacts of the complex-log mapping. Notice that the reconstructed recall is the unrotated memorized key with some noise caused by errors in the recalled phase. Figure 2d) is a histogram which graphically displays the classification vector which corresponds to $S^+ \bar{s}$. The histogram shows the interplay between the memorized images and unknown image. The "2" on the bargraph indicates which of the twelve classes the unknown object belongs. The histogram gives a value which is the best linear estimate of the image relative to the memorized objects. Another measure, the signal-to-noise ratio (SNR), is given at the bottom of the recalled image. SNR compares the variance of the ideal recall after processing with the variance of the difference between the ideal and actual recall. This is a measure of the amount of noise in the recall. The SNR does not carry much information about the quality of the recall image because the noise measured by the SNR is due to many factors such as misalignment of the center, changing reflections, and dependence between other memorized objects—each affecting quality in a variety of ways. Rotation and scale estimates are made using vector \bar{D} corresponding to the difference between the unknown vector $\tilde{\Phi}_2$ and the recalled vector $\hat{\Phi}_2$. In an ideal situation \bar{D} will be a plane whose gradient indicates the exact amount of rotation and scale the recalled object has experienced. In our system the recalled vector $\hat{\Phi}_2$ is corrupted with noise which means rotation and scale have to be estimated. The estimate is made by letting the first order difference \bar{D} at each point in the plane vote for a specified range of rotation or scale.

ORIGINAL PAGE IS
OF POOR QUALITY



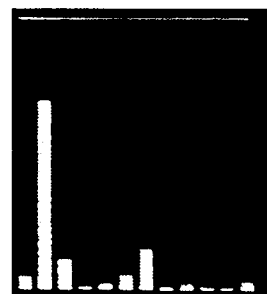
Original



Unknown



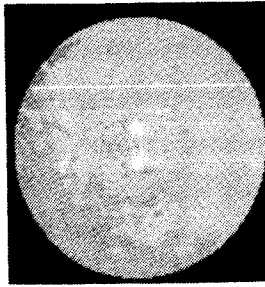
Estimated Rotation: 0°
SNR = -0.90 Db



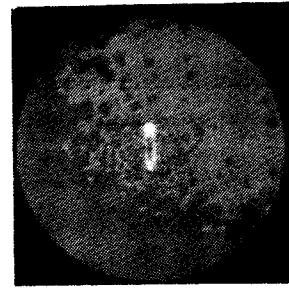
Memory:2

Figure 2: Recall Using a Scaled Key

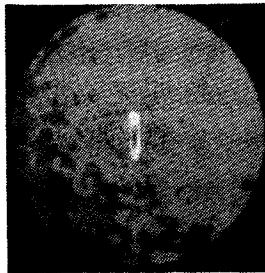
Figure 3 is the result of randomly setting the elements of the memory matrix to zero. Figure 3a) shows the ideal recall. Figure 3b) is the recall after 30 percent of the memory matrix has been set to zero. Figure 3c) is the recall for 50 percent and Figure 3d) is the recall for 75 percent. Even when 90 percent of the memory matrix has been set to zero a faint outline of the pin could still be seen in the recall. This result is important in two ways. First, it shows that the distributed associative memory is robust in the presence of noise. Second, it shows that a completely connected network is not necessary and as a consequence a scheme for data compression of the memory matrix could be found. Wechsler and Zimmerman (1988) provide additional details on the invariant recognition outlined in this section.



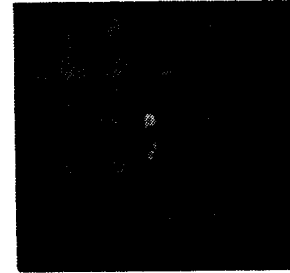
Ideal Recall



30% of Memory Set to Zero



50% of Memory Set to Zero



75% of Memory Set to Zero

Figure 3: Recall for memory Matrix Randomly Set to Zero

4.2. Sensory Integration

This section describes a series of experiments whose goal is to assess our approach to the data fusion problem. The database is made up of 15 classes of colored polyhedral objects. The objects were video taped from directly above while sitting on a flat surface. Each object class has at least one other class which presents the same shape in the image. Also several of the classes consist of objects of the same color. As discussed in section 3.3 the stimulus vectors are of the form $\bar{s}_1^T = [\bar{s}_{1R}^T \mid \bar{s}_{1B}^T \mid \bar{s}_{1G}^T]$ where R, G, B correspond to the red, green, and blue channels of the video recorder. The geometrical transformations were obtained by rotating or translating the video camera. The images used to test the system were taken at a different time than the ones used to construct this memory.

Table 1 specifies the database (DB) and the transformations the objects were subject to before the memory recall operation is performed. Each object, subject to the linear transformation specified in Table 1 is correctly recognized. The actual recall histograms are shown in Table 2. Table 3 shows the resulting histograms obtained in the presence of occlusion or noise. Approximately 1/3 of the yellow 12 sided object was erased from the image. A comparison can be made between the results of the recall with and without occlusion. A uniformly distributed noise of variance and mean equal to that of the image of the green six sided object was added. The object recall, although diminished, was still correct.

20b - 20 sided medium blue; rotation approx. 30° right
 20g - 20 sided gray; rotation 15° left
 12r - 12 sided red; smaller with rotation; very dim
 12y - 12 sided yellow; smaller
 10b - 10 sided light brown; smaller rotated 20°
 10o - 10 sided orange; rotation 40°
 8b - 8 sided light blue; rotation 30°
 8bl - 8 sided black; smaller; extremely dim
 8o - 8 sided orange; larger
 6b - 6 sided dark blue; larger
 6g - 6 sided green; rotation 45°
 6y - 6 sided yellow; larger
 4b - 4 sided dark blue; rotation 45°
 4g - 4 sided light blue; smaller rotation 10°
 4p - 4 sided lavender; rotation 30°

Table 1. - Description of 15 Classes of Colored Polyhedral
 Objects and Their Corresponding Transformations.

ORIGINAL PAGE IS
OF POOR QUALITY

10b - scaled and rotated	20b - rotated	4g - scaled and rotated	6b - scaled	8b - rotated
20b- -0.037122	20b- 0.669728	20b- 0.099462	20b- 0.121183	20b- 0.149574
20g- 0.112992	20g- 0.085861	20g- -0.005444	20g- 0.018289	20g- 0.226339
12r- 0.074970	12r- -0.063536	12r- 0.101725	12r- 0.078140	12r- 0.091095
12y- -0.002770	12y- -0.021761	12y- -0.092328	12y- -0.041012	12y- 0.033431
10b- 0.589933	10b- 0.027957	10b- 0.102195	10b- 0.082051	10b- 0.072598
10o- 0.389303	10o- 0.051344	10o- -0.020118	10o- -0.084216	10o- -0.084049
8b- -0.008726	8b- 0.038651	8b- -0.020353	8b- 0.009179	8b- 0.553082
8o- -0.038870	8o- -0.027160	8o- 0.156823	8o- -0.085697	8o- 0.277205
8b1- -0.260333	8b1- -0.206991	8b1- 0.239798	8b1- 0.274767	8b1- 0.332871
6b- -0.039565	6b- 0.189514	6b- 0.177184	6b- 0.330043	6b- -0.155819
6g- -0.011397	6g- 0.076489	6g- 0.055344	6g- 0.152556	6g- -0.043836
6y- -0.034024	6y- -0.013071	6y- 0.007695	6y- 0.040148	6y- -0.031867
4b- 0.010866	4b- 0.001625	4b- 0.112544	4b- 0.006260	4b- -0.051010
4g- 0.018852	4g- -0.000555	4g- 0.451811	4g- 0.005951	4g- 0.070805
4p- 0.081681	4p- 0.080724	4p- 0.023719	4p- 0.091934	4p- -0.201123
maximum: 10b - 0.589933	maximum: 20b - 0.669728	maximum: 4g - 0.451811	maximum: 6b - 0.330043	maximum: 8b - 0.553082
minimum: 8b1 - -0.260333	minimum: 8b1 - -0.206991	minimum: 12y - -0.092328	minimum: 8o - -0.085697	minimum: 4p - -0.201123
10o - rotated	12y - scaled	4b - rotated	6g - rotated	8b1 - scaled
20b- 0.107444	20b- -0.016374	20b- 0.033306	20b- -0.049967	20b- -0.009174
20g- 0.016014	20g- -0.030442	20g- -0.043632	20g- 0.068956	20g- -0.018269
12r- 0.019774	12r- 0.007098	12r- 0.102113	12r- -0.166670	12r- 0.046817
12y- 0.026568	12y- 0.684478	12y- -0.053745	12y- 0.003740	12y- 0.016891
10b- 0.211207	10b- 0.068539	10b- -0.021697	10b- 0.118696	10b- -0.014401
10o- 0.519179	10o- 0.002150	10o- 0.045749	10o- 0.046837	10o- 0.012176
8b- 0.014449	8b- 0.072255	8b- 0.077244	8b- -0.007574	8b- -0.000099
8o- -0.049224	8o- -0.128156	8o- 0.127757	8o- -0.077262	8o- -0.006983
8b1- -0.481672	8b1- -0.156196	8b1- 0.302109	8b1- 0.009852	8b1- 0.445833
6b- 0.266687	6b- 0.258971	6b- 0.038464	6b- 0.012869	6b- 0.099586
6g- 0.055487	6g- -0.085213	6g- -0.060469	6g- 0.461146	6g- 0.017941
6y- -0.006067	6y- 0.036109	6y- 0.057145	6y- 0.210418	6y- -0.013268
4b- 0.075286	4b- 0.044082	4b- 0.347842	4b- 0.080270	4b- 0.025530
4g- 0.061362	4g- 0.116001	4g- 0.089829	4g- 0.052528	4g- 0.020772
4p- 0.208430	4p- -0.021450	4p- 0.012316	4p- -0.078049	4p- 0.087969
maximum: 10o - 0.519179	maximum: 12y - 0.684478	maximum: 4b - 0.347842	maximum: 6g - 0.461146	maximum: 8b1 - 0.445833
minimum: 8b1 - -0.481672	minimum: 8b1 - -0.156196	minimum: 6g - -0.060469	minimum: 12r - -0.166670	minimum: 20g - 0.018269
12r - scaled and rotated	20g - rotated	4p - rotated	6y - scaled	8o - scaled
20b- -0.101214	20b- 0.074942	20b- 0.135660	20b- 0.026048	20b- 0.067367
20g- 0.061621	20g- 0.789082	20g- -0.034477	20g- -0.044931	20g- -0.056499
12r- 0.333425	12r- 0.209688	12r- -0.036622	12r- -0.226118	12r- 0.108363
12y- -0.011372	12y- -0.032170	12y- -0.020434	12y- -0.033081	12y- -0.003583
10b- 0.090512	10b- 0.009353	10b- 0.015881	10b- 0.107014	10b- -0.117320
10o- 0.107102	10o- 0.001790	10o- 0.087427	10o- 0.043923	10o- 0.009010
8b- 0.042359	8b- 0.097038	8b- -0.062108	8b- -0.078311	8b- -0.051742
8o- -0.090681	8o- 0.113085	8o- 0.044841	8o- 0.035680	8o- 0.986751
8b1- 0.138120	8b1- 0.083690	8b1- 0.134441	8b1- -0.225961	8b1- -0.184686
6b- 0.119825	6b- 0.187161	6b- 0.063891	6b- 0.066302	6b- -0.202276
6g- -0.071757	6g- -0.066753	6g- 0.063415	6g- 0.645227	6g- 0.031471
6y- 0.077901	6y- 0.073837	6y- -0.022305	6y- 0.683283	6y- 0.017449
4b- -0.016150	4b- -0.145125	4b- 0.059183	4b- 0.088817	4b- 0.034253
4g- 0.032028	4g- -0.046533	4g- 0.042149	4g- 0.024370	4g- 0.124651
4p- 0.053942	4p- -0.283062	4p- 0.329339	4p- 0.032338	4p- 0.371463
maximum: 12r - 0.333425	maximum: 20g - 0.789082	maximum: 4p - 0.329339	maximum: 6y - 0.683283	maximum: 8o - 0.986751
minimum: 20b - -0.101214	minimum: 4p - -0.283062	minimum: 8b - -0.062108	minimum: 12r - -0.226118	minimum: 6b - -0.202276

Table 2. Responses of the Fusion Memory

12y - scaled

20b= -0.016374
 20g= -0.030442
 12r= 0.007098
 12y= 0.684478
 10b= 0.068539
 10o= 0.002150
 8b= 0.072255
 8o= -0.128156
 8bl= -0.156196
 6b= 0.258971
 6g= -0.085213
 6y= 0.036109
 4b= 0.044082
 4g= 0.116001
 4p= -0.021450
 maximum: 12y = 0.684478
 minimum: 8bl = -0.156196

20b= -0.061517
 20g= 0.055189
 12r= 0.036854
 12y= 0.638085
 10b= 0.084086
 10o= 0.059582
 8b= 0.086944
 8o= -0.140270
 8bl= -0.566859
 6b= 0.114629
 6g= -0.098479
 6y= 0.043607
 4b= 0.118768
 4g= 0.107182
 4p= 0.100319
 maximum: 12y = 0.638085
 minimum: 8bl = -0.566859

without occlusion

with occlusion

6g - rotated

20b= -0.049967
 20g= 0.068956
 12r= -0.166670
 12y= 0.003740
 10b= 0.118696
 10o= 0.046837
 8b= -0.007574
 8o= -0.077262
 8bl= 0.009852
 6b= 0.012869
 6g= 0.461146
 6y= 0.210418
 4b= 0.080270
 4g= 0.052528
 4p= -0.078049
 maximum: 6g = 0.461146
 minimum: 12r = -0.166670

20b= -0.039071
 20g= 0.098584
 12r= -0.141735
 12y= -0.008002
 10b= 0.110771
 10o= 0.091265
 8b= -0.051865
 8o= -0.045028
 8bl= 0.070492
 6b= -0.042376
 6g= 0.413561
 6y= 0.204100
 4b= 0.077749
 4g= 0.093187
 4p= -0.097903
 maximum: 6g = 0.413561
 minimum: 12r = -0.141735

without noise

noise (0Db,mean=5)

Table 3. Invariant Recognition to Occlusion and Noise

The next series of experiments was aimed at exploring the capability of our system when exposed to overlap situations, like those encountered during bin-picking. The database for this memory consists of six objects. Three of the objects have a side view learned state along with the standard top-down view used in the previous example. This memory is tested using three objects which overlap (the central object is supported by the other two). The camera moves, similar to a conveyor belt, and samples the image in five distinct locations. Figure 4 shows several graphs which indicate how the recall histogram varies as the camera is moved. The top three graphs show the response of the three objects which were used in the test. Each of these objects were memorized in two views—one from the top looking down and a second view from the side. The dotted line on all of the graphs shows the maximum response given by any object in the memory. The results show that for points close to the central locations of the objects being viewed the response is correct. For viewpoints between two objects the response was dominated by one of the objects present in the input. The drawing 4e) shows the placement of the sampled center points.

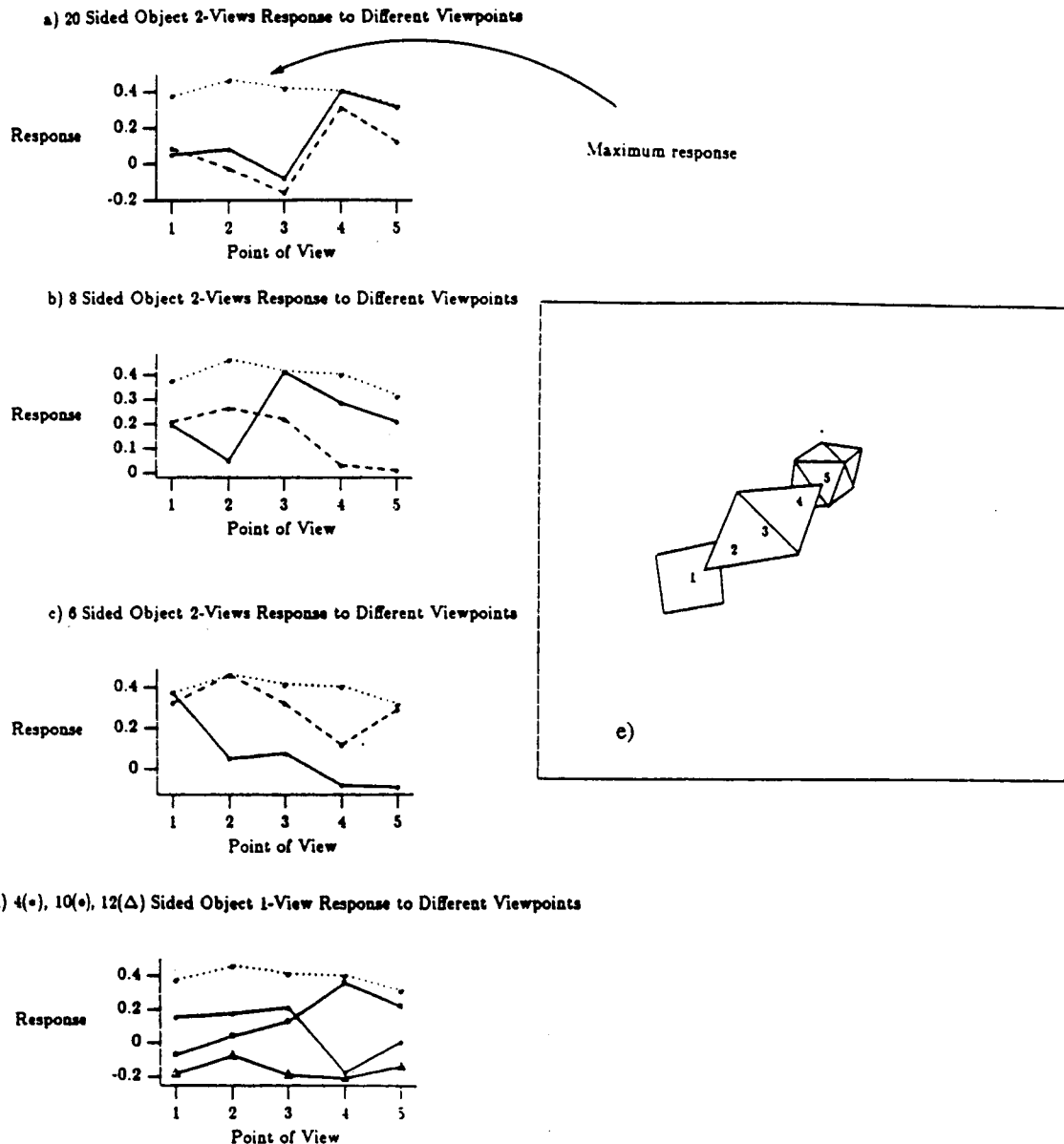


Figure 4 Response of memory at multiple viewpoints for the overlapped configuration

5. Conclusion

Our experiments demonstrate the feasibility of our system for 2-dimensional invariant recognition and data fusion. Information, implicit in the input patterns, is made explicit through the preprocessing subsystem. This is followed by the distributed associative memory which stores and retrieves information. The DAM, because of its distributed nature, lends robustness to the overall system function in the presence of noise, occlusion, and other factors which degrade performance. There are some properties of the DAM which are crucial for data integration. In the DAM, the parameters necessary for combining multiple sources are extracted from the environment and recall is context sensitive. We feel an important aspect of our system is the natural way it processes iconic information which is in contrast to AI symbolic approaches.

There are implicit weaknesses in the Neural Network model we have chosen for the heart of the recognition system. The distributed associative memory we use is linear, and as a result there are certain desirable properties which will not be exhibited by our computer vision system. For example, feedback through our system will not improve recall from the memory. Recall could be improved if a non-linear element, such as a sigmoid function, is introduced into the feedback loop. Non-linear neural networks, such as those proposed by Hopfield (1982) or Anderson et al. (1977), can achieve this type of improvement because each memorized pattern is associated with stable points in an energy space. The price to be paid for the introduction of non-linearities into a memory system is that the system will be difficult to analyze and can be unstable. Implementing our computer vision system using non-linear distributed associative memory is a goal of our future research.

The DAM, like most other NN models, does not exploit the topographical power naturally present in input visual information. Examinations of the visual cortex have shown that visual information is processed through multiple mappings of the visual fields. Incorporation of this type of information into a NN model could lead to dramatic compression of the necessary number of connections and more processing power for visual tasks. All pattern recognition techniques, including NN, are sensitive to the nature of the training set. If the training set is not representative of the classes which will be encountered in the environment then the abilities of the system to classify and generalize will be hampered. Research in the area of NN is in its infancy. We expect the future to bring better understanding of these characteristics along with new methods of learning classes.

The computer vision system presented in this paper was designed with only 2-dimensional metric distortions in mind. We have shown some ability to deal with clustered 3-dimensional polyhedra as presented in the previous experiments. We are presently extending our work toward 3-dimensional object recognition. We propose to use an approach based on characteristic views. (Chakravarty and Freeman, 1982) or aspects (Koenderink and Van Doorn, 1979) which suggests that the infinite 2-dimensional projections of a 3-dimensional object can be grouped into a finite number of topological equivalence classes. An efficient 3-dimensional recognition system would require a parallel indexing method to search for object models in the presence of geometric distortions, noise, and occlusion. Our object recognition system using distributed associative memory can fulfill those requirements with respect to characteristic views. The strength of biological vision and the weaknesses in computational vision when it comes to analyzing, sensing, and integrating information from the environment, suggest that certain aspects need to be incorporated into future systems. First, the interpretation of the environment should be done in 3-dimensions because the world is 3-dimensional. Second, maintaining spatial integrity of the scene and its

projection is extremely important for recognition and integration. Third, the integration of information should take place in an active manner. Finally, and most importantly, methods of integration need to include methods for verifying assumptions and learning from the environment.

References

- [1] Anderson, C. H. P. J. Burt, and G. S. Van Der Wal (1985), Change detection and tracking using pyramid transform techniques, Proc. of the SPIE Conf. on Intelligent Robots, and Computer Vision, Vol. 579, 72-78.
- [2] Anderson, J. A., J. W. Silverstein, S. A. Ritz, and R. S. Jones (1977), Distinctive features, categorical perception, and probability learning: some applications of a neural model, Psychol. Rev., 84, 413-451.
- [3] Biederman, I. (1987), Recognition-by-components: A theory of human image understanding, Psychological Review, vol. 94, no. 2, 115-147.
- [4] Caulfield, H. J. and M. H. Weinberg (1982), Computer recognition of 2-D pattern using generalized matched filters, Applied Optics, 21, 9.
- [5] Chakravarty, I., and H. Freeman (1982), Characteristic views as a basis for 3-D object recognition, Proc. SPIE on Robot Vision, 336, 37-45.
- [6] Faugeras, O. D., and M. Hebert (1986), The representation, recognition, and positioning of 3-D shapes from range data, in Techniques for 3-D Machine Perception, A. Rosenfeld (Ed.), North-Holland, 13-52.
- [7] Garvey, T. D., and J. D. Lowrance (1983), Evidential reasoning: an implementation for multisensor integration, TN 307, AI Center, SRI, Palo Alto, CA.
- [8] Gordon, J., and E. H. Shortliffe (1984), The Dempster-Shafer theory of evidence, in Rule-Based Expert Systems, Buchanan, B. G. and E. H. Shortliffe (eds.), Addison-Wesley.
- [9] Hartline, P. L. Kass, and M. Loop (1978), Merging of modalities in the optic tectum: infrared and visual integration in the rattlesnake, Science 199, 545-548.
- [10] Hebb, D. O. (1949), The Organization of Behavior, New York: Wiley.
- [11] Hecht-Nielsen, R. (1986), Artificial neural system technology, TRW AI Center.
- [12] Hester, C., and D. Casasent (1981), Interclass discrimination using synthetic discriminant functions (SDF), Proc. SPIE on Infrared Technology for Detection and Classification, 302.
- [13] Hopfield, J. J. (1982), Neural networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci. USA, 79, April 1982.
- [14] Julesz, B. (1975), Experiments in the visual perception of texture, Scientific American, 232, 34-43.
- [15] Knudsen, E. I. and M. Konishi (1978), A neural map of auditory space in the owl, Science 200, 795, 795-797.
- [16] Knudsen, E. I. (1982), Auditory and visual maps of space in the optic tectum of the owl, J. of Neuroscience 2, 1177-1194.
- [17] Koenderink, J. J., and A. J. Van Doorn (1979), Internal representation of solid shape with respect to vision, Biological Cybernetics, 32, 4, 211-216.
- [18] Kohonen, T. (1984), Self-Organization and Associative-Memories, Springer—Verlag.
- [19] Marr, D., Vision, W. H. Freeman, 1982.

- [20] Massone, L., G. Sandini, and V. Tagliasco (1985), "Form-invariant" topological mapping strategy for 2D shape recognition, CVGIP, 30, 169-188.
- [21] McClelland, J. L., D. E. Rumelhart, and the PDP Research Group (Eds.) (1986), Parallel Distributed Processing, (Vol. 1, 2), MIT Press.
- [22] Poggio, T. (1985), Early vision: From computational structure to algorithms and parallel hardware, Computer Vision, Graphics and Image Processing, 31, 139-155.
- [23] Simon, H. A. (1982), The Sciences of the Artificial (2nd ed.), MIT Press.
- [24] Stiles, G. S. and D. L. Denq (1985), On the effect of noise on the Moore-Penrose generalized inverse associate memory, IEE Trans. on PAMI, 7, 3, 358-360.
- [25] Van Essen, D. C., and J. H. R. Maunsell (1983), Hierarchical organization and functional streams in the visual cortex, TINS (Trends in Neuro Science).
- [26] Waxman, A. M., and J. H. Duncan (1986), Binocular image flows: steps toward stereo motion fusion, IEEE Trans. on PAMI, 6, 715-729.
- [27] Wechsler, H. and G. L. Zimmerman (1988), 2-D invariant object recognition using distributed associative memory, IEEE Trans. on Pattern Analysis and Machine Intelligence (to appear).
- [28] Yonas, A., M. E. Arterberry, and C. E. Granrud (1987), Space perception in infancy, Annals of Child Development, 4, 1-34.

RANGE DATA DESCRIPTION BASED ON MULTIPLE CHARACTERISTICS

EZZET AL-HUJAZI
WAYNE STATE UNIVERSITY, DETROIT, MICHIGAN

ARUN SOOD
GEORGE MASON UNIVERSITY, FAIRFAX, VIRGINIA

ABSTRACT

An algorithm for describing range images based on Mean curvature (H) and Gaussian curvature (K) is presented in this paper. Range images are unique in that they directly approximate the physical surfaces of a real world 3-D scene. The curvature parameters are derived from the fundamental theorems of differential geometry and provides visible invariant pixel labels that can be used to characterize the scene. The sign of H and K can be used to classify each pixel into one of eight possible surface types. Due to the sensitivity of these parameters to noise the resulting HK -sign map does not directly identify surfaces in the range images and must be further processed. In this paper a region growing algorithm based on modeling the scene points with a Markov Random Field (MRF) of variable neighborhood size and edge models is suggested. This approach allows the integration of information from multiple characteristics in an efficient way. The performance of the proposed algorithm on a number of synthetic and real range images is discussed .

PRECEDING PAGE BLANK NOT FILMED

1 INTRODUCTION

This paper describes an algorithm for 3-D object description. The range data of the points on the visible surface are available from a scanning laser range finder. To generate a useful description, a representation is needed. Surface description is one of the most widely used methods of object description. It offers a rich, stable description, has local support so that partially visible objects can be identified and it enables us to recreate a shape reasonably close to the original one.

In differential geometry the information given by the sign of H and K can be used to classify a surface point into one of eight possible labels. These two surface curvatures are derived from the first and second fundamental forms. They are sensitive to noise and the resulting HK-sign map does not correspond directly to surfaces in the image and thus it has to be further processed. In this paper an algorithm based on MRF and edge models is suggested for processing the HK-sign map. This approach is chosen because it allows an analytical basis for integrating a number of object features. Further, the use of variable neighborhood area provides a good compromise between the speed of processing and the number of pixels misclassified by the algorithm.

The next section presents a review of previous approaches to surface description. Section 3 presents a review of relevant differential geometry results, and Section 4 presents a review of MRF and the Gibbs Distribution (GD). Our algorithm will be given in Section 5. Section 6 shows results of processing various range images and Section 7 outlines the conclusions.

2 APPROACHES TO SURFACE DESCRIPTION

The approaches to 3-D surface description can be divided into 1) Approximation by simple surfaces, 2) edge extraction, and 3) 3-D surface characterization.

2.1 Simple Surfaces

Surface approximation using simple surface patches is an important surface representation approach in computer graphics. The earliest work used approximation by planar patches. Later methods have used other surface patches such as 2-D splines. The number of patches found by this method is typically very large and the points and lines, where the approximating patches are joined, need not have any significance. Some examples of using surface approximation can be found in [5,10].

2.2 Edge Extraction

The jump boundaries can be easily located by using specialized operators similar to those used for intensity edge detection. The detection of fold edges is a more difficult problem. Some successful methods have been developed for the detection of these features [6,9].

2.3 3-D Surface Characterization

A surface characteristic is a descriptive feature of a general smooth surface. Surface characterization refers to the computational process of partitioning surfaces into regions with similar characteristics. The descriptive quality of the features used to identify surfaces is of critical importance to the surface description process. The methods to characterize surfaces can be grouped into two sub-classes. The first describes the surfaces by point wise properties, whereas the second attempts to derive global descriptions. Some examples of using surface characterization can be found in [1,2,3,11].

3 H AND K CURVATURE PARAMETERS

H and K are identified as the local second order surface characteristics that possess several invariance properties and represent extrinsic and intrinsic surface geometry features respectively. The sign of these surface curvatures can be used to classify the image surface points into one of eight basic types. Fig. (1) shows the corresponding surfaces labels. These two curvature parameters can

be calculated using [1] :

$$K = \frac{f_{xx}f_{yy} - f_{xy}^2}{(1+f_x^2+f_y^2)^{3/2}}, \quad H = \frac{f_{xx}+f_{yy}+f_{xx}f_y^2+f_{yy}f_x^2-2f_xf_yf_{xy}}{(1+f_x^2+f_y^2)^{3/2}}$$

	K>0	K=0	K<0
H<0	Peak	Ridge	Saddle Ridge
H=0	---	Flat	Minimal surface
H>0	Pit	Valley	Saddle Valley

Fig.(1) Surface type labels from surface curvature sign.

Some of the problems with the HK-sign map are :a) Preliminary smoothing is necessary to obtain reasonable values for H and K [1]. However, after filtering the HK-sign surface labels then reflects the geometry of the smoothed surface data. Hence, the HK-sign map must be further processed, b) In the presence of noise HK-sign map surface labels tend to connect the labels of neighborhood, but distinct, surface regions. c) Global surface properties is lacking.

4 MRF AND THE GD

The concept of a MRF is a direct extension of the concept of a Markov process to higher dimension [7]. A discrete MRF on a finite lattice is defined as a collection of random variables, which correspond to the sites of the lattice.

Definition of MRF:

Let $N_1 \times N_2$ correspond to a rectangular lattice defined as:

$L = \{ (i,j) : 1 \leq i \leq N_1, 1 \leq j \leq N_2 \}$. A collection of subsets of L defined as:

$q = \{ q_{ij} : (i,j) \in L, q_{ij} \subseteq L \}$ is a neighborhood system on L iff;

a- $(i,j) \notin q_{ij}$, and b-if $(k,l) \in q_{ij}$ then $(i,j) \in q_{kl}$ for any $(i,j) \in L$.

A random field $X = \{X_{ij}\}$ defined over a lattice L is a MRF with respect to the neighborhood system q iff:

$$P(X_{ij}=x_{ij}|X_{kl}=x_{kl}, (k,l) \in L, (k,l) \neq (i,j))=P(X_{ij}=x_{ij}|X_{kl}=x_{kl}, (k,l) \in q_{ij})$$

for all $(i,j) \in L$, and $P(X=x)>0$ for all x .

A major difficulty in applying MRF formulation is in the definition of a valid conditional distribution. An alternative approach is based on the Markov-Gibbs equivalence established by the Hammersley-Clifford theorem. Before stating the theorem the neighborhood system and the GD are defined.

Definition of cliques:

Given a system of neighborhoods on a lattice a clique c of the pair (L,q) is a subset of L such that;

- a) c consists of a single pixel, or
- b) for $(i,j) \neq (k,l)$, $(i,j) \in c$ and $(k,l) \in c$ implies that $(i,j) \in q_{kl}$. The collection of all cliques of (L,q) is denoted by $C=c(L,q)$. The neighborhood system and the cliques associated with the first order system are shown in Fig.(2).

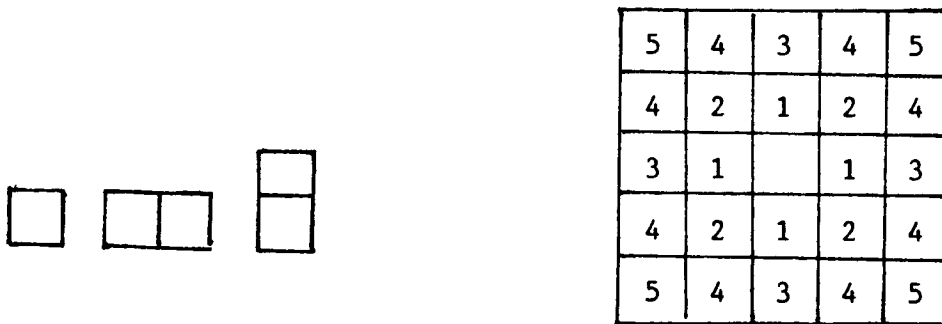


Fig.(2) The neighborhood systems and the cliques for the first order.

Definition of GD:

A random field $X=\{x_{ij}\}$ defined on L has an associated GD (or equivalently is a Gibbs Random Field(GRF) with respect to q) iff its joint distribution is of the form:

$$P(X=x) =(1/Z)*\exp(-U(x))$$

where $U(x) = \sum_c V_c(x)$ is the energy function,

V_c = potential associated with clique c ,

and $Z = \sum_x \exp(-U(x))$ is a normalization factor.

Hammersley-Clifford theorem:-Let q be a neighborhood system on a finite lattice L . A random field X is a MRF with respect to q iff its joint distribution is a GD with cliques associated with q .

5 OUR ALGORITHM

Biological vision systems achieve efficient, robust and reliable recognition in highly variable environments through the integration of many visual sources. For example the simple task of locating objects boundaries can be performed far more effectively by integrating evidence of discontinuities in image intensity, stereo disparity, speed and direction of motion and texture information than by using evidence from a single visual source on its own. The integration problem is computationally complex. The integration can be achieved by associating a MRF on a lattice to each physical process and another (binary) model to its discontinuities. The lattices are coupled to each other to reflect the interdependence of the corresponding process in image formation. Similar work using this approach can be found in [4,8]. In general, the latter methods, are computationally expensive and the number of quantization levels must be small (typically 2 or 3). The use of H and K allows us to reduce the number of levels from 256 (the original image) to 3 levels ($-$, 0 , $+$ for H and K).

The flow chart of our algorithm is shown in Fig. (3). The H and K are calculated in multi-scale fashion. Then the output of the multi-scale is combined with the edge information and the surface normals. This will give us a seed region and edge information which will be entered to the region growing algorithm. H and K are processed separately and then combined to obtain the HK-sign map. To obtain the final surface description of the object, surfaces are fitted to the HK-sign map.

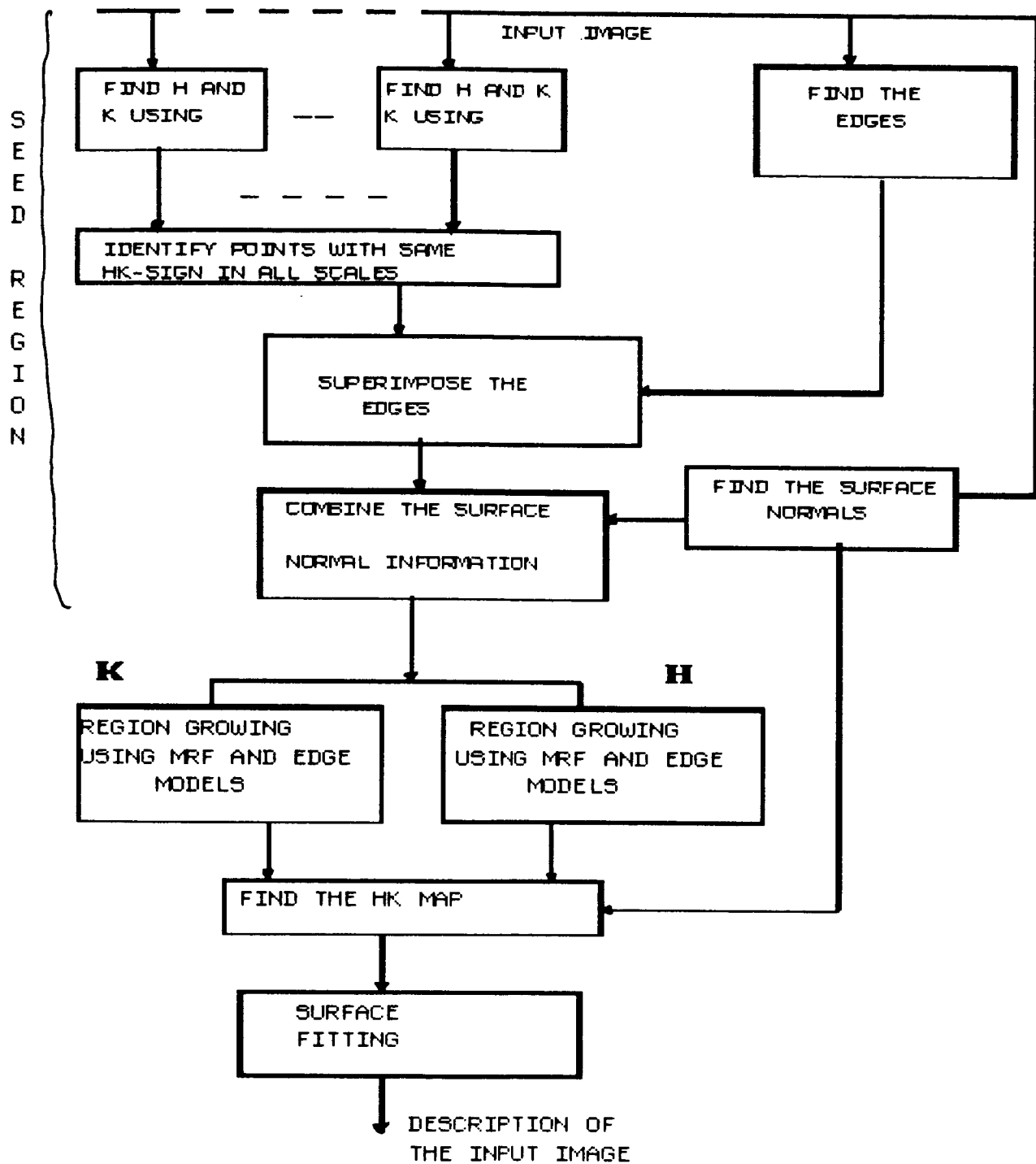


Fig. (3) The algorithm flow chart.

5.1 Finding the Seed Region:

The seed regions are obtained using a multi-scale approach. This approach is justified because the output from different scales is going to change significantly on the boundary of the object while the points well inside the surface will not change. The input image is smoothed with a Gaussian filter of different standard deviation and for each output the values of H and K are estimated. The sign of the resulting H and K values are then used to form a three level image for H and K. The outputs from the multi-scale are then combined by identifying the points on the different scales (3 in our experiments) where the value of H and K signs are identical. The labeling of these points are assumed to be correct and is used as the seed region for the region growing algorithm. The edges are also obtained and superimposed on the the multi-scale output.

In some cases, for example for a roof edge, surface normal information is needed to segment the planar regions. The surface normal is estimated by fitting a plane in a 3X3 mask size. The surface normal information is also used to segment the background of the object.

5.2 Region Growing:

The seed region output is entered to the region growing step. The region points are modeled as a MRF with variable neighborhood size. For the edge points, an additional term is used. The energy function integrates these two models :

$$U(f,e;g)=\sum_q V(e)+(1-e)*\sum_q V(f_i,f_j|e)$$

where g is the output from the seed region step, e is the edge point (binary), f is the processed image, q is the neighborhood system, V(e) is the energy function due to the presence of edge. V(e) is computed by using Fig.(4) in which a value is assigned for V(e) based on all the possible local configurations of the edge point. This model encourages the formation of continuous edges and discourages thick edges. For example if points B and C are edge points the model discourage the presence of edge at A.

$V(f_i, f_j | e)$ is the energy function due to the pixel label in the neighborhood area given the edge points. Only the single pixel clique is used in the experimental results.

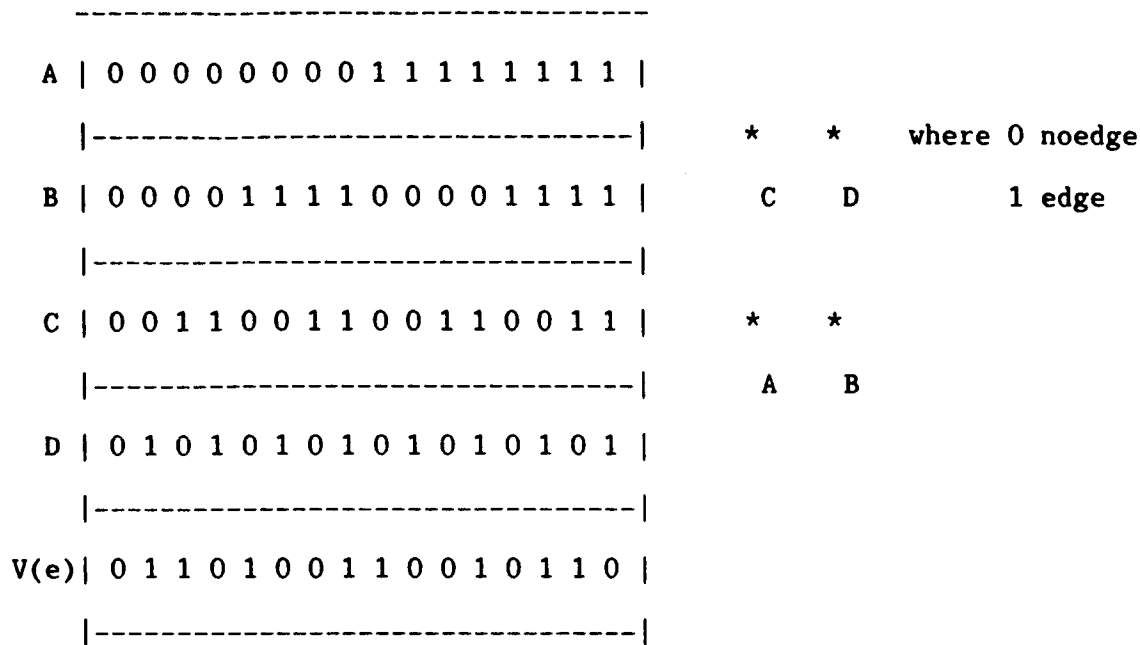


Fig.(4) The Edge Model

The region growing algorithm proceeds by collecting the edge points and the pixels unclassified by the multi-scale approach in an array. A point is then picked at random. The energy function given earlier is then minimized in the local area surrounding the selected pixel. This is repeated for all the points in the array for a number of iterations (maximum of 30 was used in the experimental results).

6 EXPERIMENTAL RESULTS

The algorithm has a good parallel computational structure, since the multi-scale, edge detection and the surface normal estimation can be computed simultaneously. Also the computation of H and K are independent and can be computed simultaneously. The algorithm has been tested on a number of synthetic and real

images. The images are 128X128 with 8 bits/pixel. The H and K values are obtained following the procedure suggested by [1]. Experimental results for different objects are shown in Fig.(5) through (7). To assess the importance of the edge information, images are processed with and without the edge model. The neighborhood system of the region model has been varied from first to fifth order system.

The first object (Fig.(5a)) is a synthetic image of a sphere. The output of the seed region step is shown in Fig.(5b) for H and in Fig.(5e) for K. The result of the region growing step is shown in Fig.(5c) for H and in Fig.(5d) for K. Fig.(5f) shows the final HK-sign map obtained by combining the output from Fig.(5c) and Fig.(5d). As can be seen the image is segmented perfectly using this method. Fig.(6a) is a range image of a coke bottle obtained using a laser range finder at the Environmental Research Institute of Michigan (ERIM). The content of Fig.(6) are similar to Fig.(5). Good segmentation is obtained with the exception of a small area at the tip of the coke bottle.

Fig.(7) shows the results for a coffee cup obtained from ERIM. Fig.(7a) shows the image. Fig.(7d) and Fig.(7h) show the seed region obtained for H and K respectively. The range image is then processed in two different ways. Fig.(7e) and Fig.(7i) show the output of the region growing algorithm with the edge model. Fig.(7b) shows the final HK-sign map. The segmentation results obtained were good with the exception of the handle of the coffee cup, which was not classified. This is because of the size of this region and the restriction in the algorithm on the number of pixels required for classification. In Fig.(7f) and Fig.(7g) the outputs of the region growing algorithm without the edge model are shown. Fig.(7c) shows the final HK-sign-map. In this case the handle is classified as planar region, also small regions of the cylindrical surfaces of the object are classified as planar. A comparative study of Fig.(7b) and Fig.(7c) illustrates that inclusion of the edge model leads to less misclassified points. To emphasize the advantage of using a variable neighborhood system for the MRF, Fig.(8) shows the results of processing the coffee cup with different fixed neighborhood

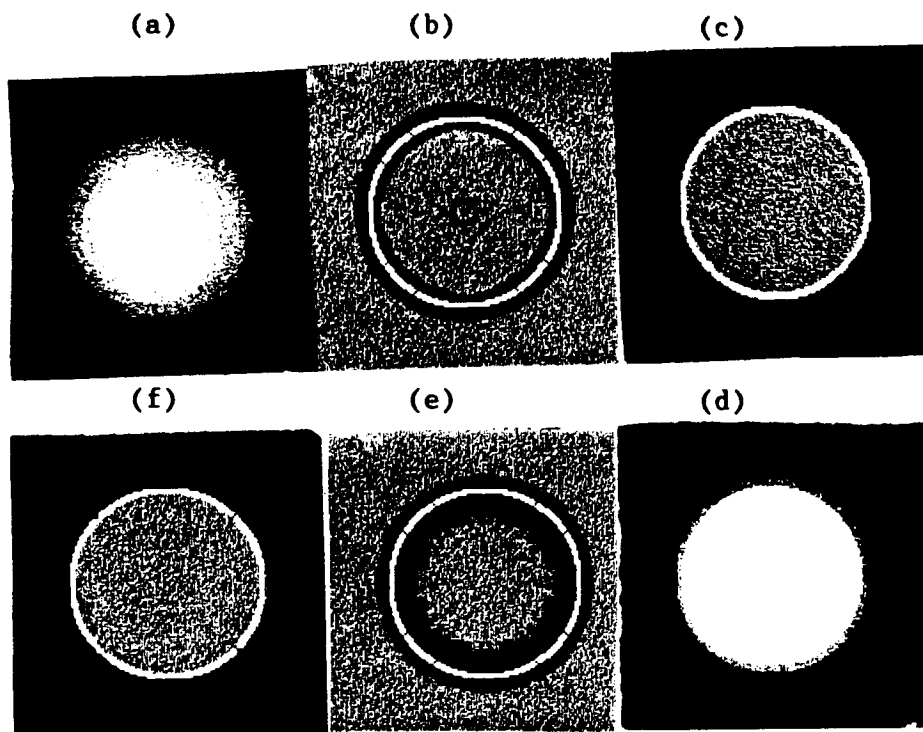


Fig.(5) Results of Processing a Synthetic Image -A sphere.

- a)The original range image.
 - b)H seed region multi-scale output (Blach:Unclassified).
 - c)H region growing output (Gray:H<0;Black:H=0).
 - e)K seed region multi-scale output (Black: unclassified).
 - d)K region growing output (white:K>0;Black:K=0).
 - f)HK-sign map (Black:planar surafce;Gray: peak surface).
- Edges are superimposed on Fig.5b through 5f.

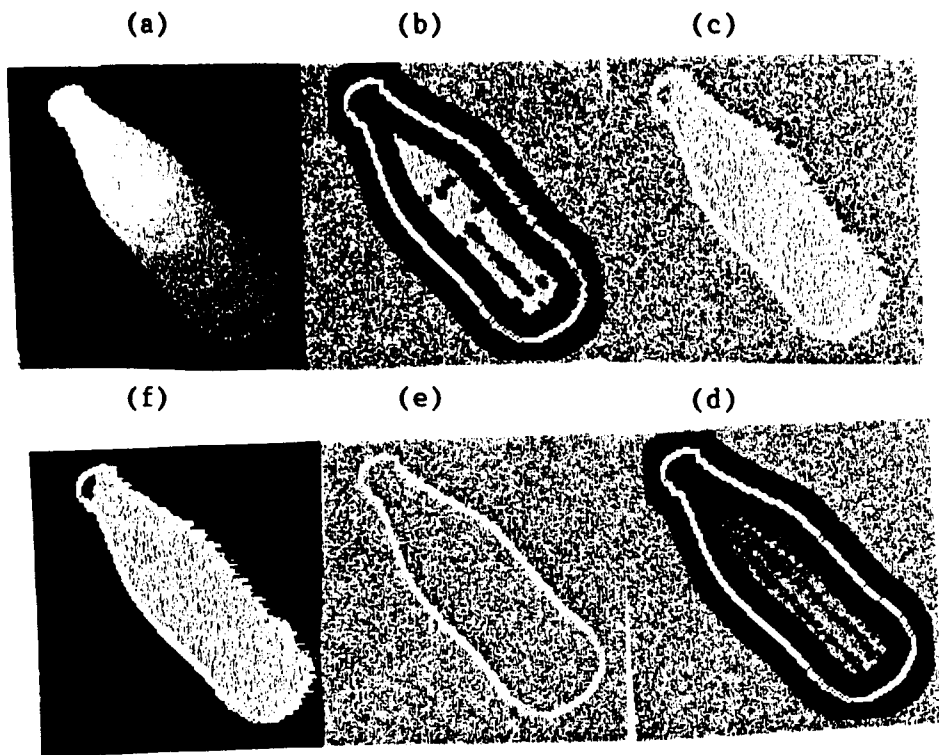


Fig.(6) Results of Processing a Synthetic Image -A Coke Bottle.

- a)The original range image.
 - b)H seed region multi-scale output (Black:Unclassified).
 - c)H region growing output (Bottle: $H < 0$; Background: $H = 0$).
 - d)K seed region multi-scale output (Black:unclassified).
 - e)K region growing output ($k = 0$).
 - f)HK-sign map (Black:planar surface;Gray:Ridge surface).
- Edges are superimposed on Fig.5b through 5f.

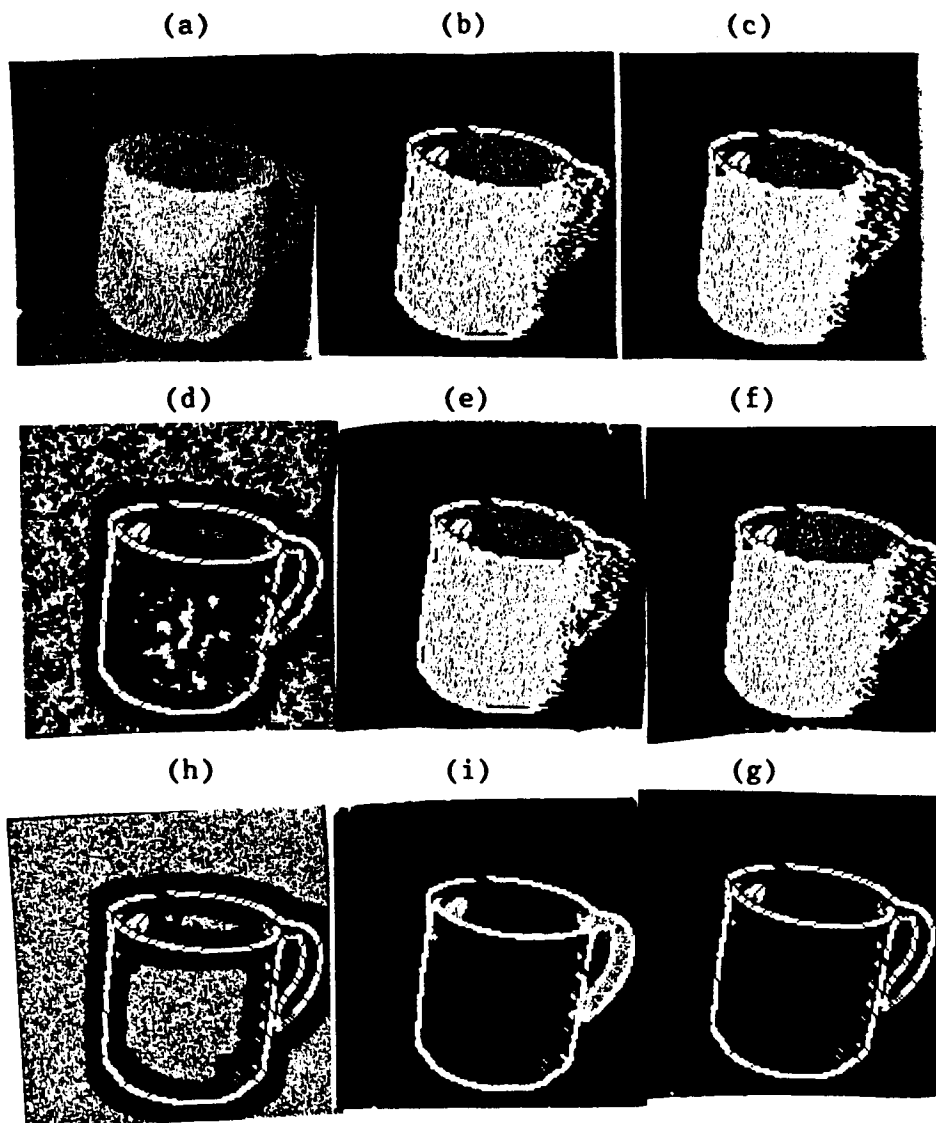


Fig.(7) Results of processing a range image -A Coffee Cup.

- a)The original image.
 - b)The HK-map with edge model (Gray:Ridge surface;Black:Planar surface; White:Valley Ridge).
 - c)Similar to (b) without edge model.
 - d)H seed region multi-scale output (Black:Unclassified).
 - e)H region growing output with edge model (Gray:H>0;Black:H=0; White:H<0;Handle Unclassified).
 - f)Similar to (e) without edge model (Handle classified as planar).
 - h)K seed region multi-scale output (Black:Unclassified).
 - i)K region growing output with edge model (Black:K=0;Gray:Unclassified).
 - g)Similar to (i) without edge model (Black:Planar).
- Edges are superimposed on Fig.7b through 7g.

systems. In this figure the time required for processing is compared for five different neighborhood systems. The time required for the variable neighborhood system (up to the fifth order) is also shown. The other graph in the figure shows the difference in the classification between the variable and the fixed neighborhood system MRF for a fixed number of iterations (30). Thus the use of the variable neighborhood system gives a good compromise between the time required for processing and the number of misclassified pixels.

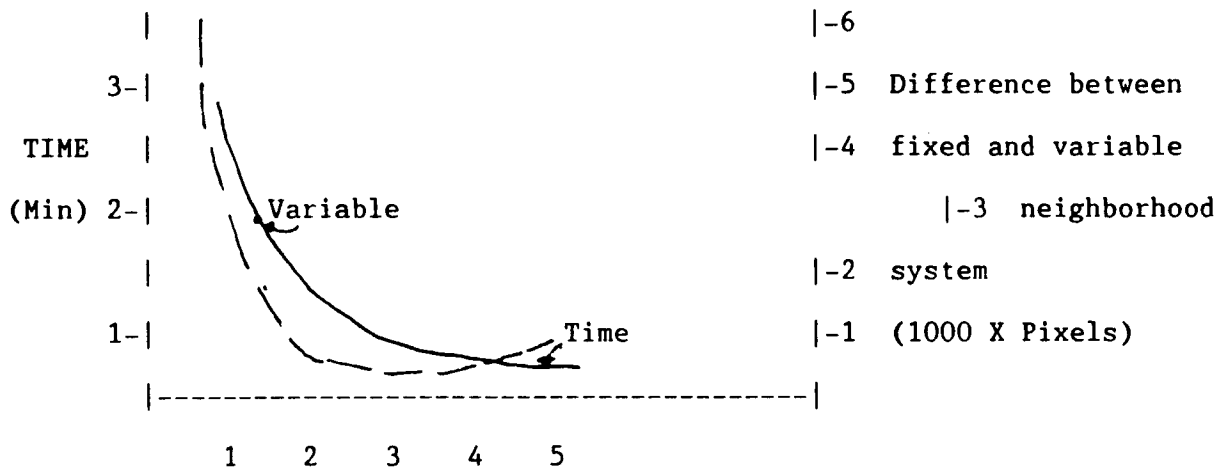


Fig.(8) Comparison between fixed and variable neighborhood system
a)Time,b)Difference in classification.

7 CONCLUSION

An algorithm for segmentating range images using a variable neighborhood system MRF and edge models is presented. This approach allows us to integrate a number of surface characteristics in an efficient way. The results shown are good with the exception of the handle of the coffee cup. The use of H and K allow us to work with a small number of levels (3 compared with 256) which makes the processing faster. The use of variable neighborhood system MRF reduces the number of misclassified pixels with a small increase in the time required for processing.

REFERENCES

- [1] P.J. Besl, "Surfaces in Early Range Image Understanding," Ph.D. Dissertation, Dep. Elec. Eng. Comput. Sci., Univ. of Michigan, Ann Arbor, Rep. RSD-TR-10-86, Mar. 1986.
- [2] M. Brady, J. Ponce, A. Yuille and H. Asada, "Describing Surfaces," Comput. Vision. Graphics, Image processing, vol. 32, pp. 1-28, 1985.
- [3] T.G. Fan, G. Medioni, and R. Nevatia, "Description of Surfaces From Range Data Using Curvature Properties," In Proc. Computer Vision and Pattern Recognition Conf., IEEE Comput. Soc., Miami, FL, June 22-26, pp. 86-91, 1986.
- [4] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distribution, and Bayes Restoration of Images," IEEE Trans. Pattern Anal. Machine Intell., vol.6, no.6, pp. 721-741, Nov. 1984.
- [5] T.C. Henderson, "Efficient 3-D Object Representations for Industrial Vision Systems," IEEE Trans. Pattern Anal. Machine Intell., vol.5, no.6, pp. 609-617, Nov. 1983.
- [6] S. Inokuchi, T. Nita, F. Matsuday, and Y. Sakurai, "A Three-Dimensional Edge-Region Operator for Range Pictures," In Proc. 6th Int. Conf. Pattern Recognition, Munich, West Germany, pp. 918-920, Oct. 19-22, 1982.
- [7] R. Kindermann and J.L. Snell, "Markov Random Fields and their Applications," vol.1, Amer. Math. Soc.(1980).
- [8] J.L. Marroquin, "Probabilistic Solution of Inverse Problems," M.I.T. Artificial Intelligence Lab., Cambridge, MA, Tec. Rep. 860, 1985.
- [9] A. Mitiche and J.K. Aggarwal, "Detection of Edges Using Range Information," IEEE Trans. Pattern Anal. Machine Intell., vol.5, no.2, pp. 174-178, 1983.
- [10] M. Oshima and Y. Shirai, "Object Recognition Using Three Dimensional Information," IEEE Trans. Pattern Anal. Machine Intell., vol.5, no.4, pp. 353-361, July 1983.
- [11] J. Ponce and M. Brady, "Toward a Surface Primal Sketch," In Proc. IEEE Int. Conf. on Robotics and Automation, pp. 420-425, St. Louis, MO, March 25-28 1985.

Data Management

**The Second Generation Intelligent User Interface For The
Crustal Dynamics Data Information System**

**Spacelab Data Processing Facility Quality Assurance/Data
Accounting Expert Systems: Transition From Prototypes
To Operational Systems**

**Automated Cataloging And Characterization Of Space
Derived Data**

A Design For A Ground-Based Data Management System

PRECEDING PAGE BLANK NOT FILMED

THE SECOND GENERATION INTELLIGENT USER INTERFACE FOR THE CRUSTAL DYNAMICS DATA INFORMATION SYSTEM

Nicholas Short Jr. Code 634
NASA/Goddard Space Flight Center
National Space Science Data Center
Greenbelt, Md. 20771

Scott L. Wattawa
Science Application Research Inc.
National Space Science Data Center
Greenbelt, Md. 20771

1. INTRODUCTION

For the past decade, operations and research projects that support a major portion of NASA's overall mission have experienced a dramatic increase in the volume of generated data and resultant information that is unparalleled in the agency's history. The effect of such an increase is that most of the science and engineering disciplines are undergoing an information glut, which has occurred, not only because of the amount, but also because of the type of data being collected (e.g., spatial data).

This information glut is growing exponentially and is expected to grow for the foreseeable future. Consequently, it is becoming physically and intellectually impossible to identify, access, modify, and analyze the most suitable information. Thus, the dilemma arises that the amount and complexity of information has exceeded and will continue to exceed, using present information systems, the ability of all the scientists and engineers to understand and take advantage of this information.[Kneale88] [Gao87]

As a result of this information problem, NASA has initiated the Intelligent Data Management (IDM) Project to design and develop Advanced Information Management Systems (AIMS). The first effort of the Project was the prototyping of an Intelligent User Interface (IUI) to an operational scientific database using expert systems, natural language processing, and graphics technologies. This paper presents an overview of the IUI formulation and development for the second phase.

2. PROBLEMS WITH EXISTING DATABASE SYSTEMS AND INTERFACES

Present database systems have mainly been designed and developed for one specific purpose: to efficiently archive and manage data for a specific domain by and, in many times, only for developers with a background in computer science and related fields.[Bic86] Therefore, most database systems, in practice, suffer from the intrinsic flaw of not effectively servicing the casual or new user. The reason for this defect is that the context or pragmatics of the user's understanding and language cannot be expressed in the data and data structures. The specific problems that result in the above design inadequacy are:

- databases are designed for efficient storage and query operations by a database designer who usually has little or no understanding of the database's application domain; [Bic86]
- databases have limited capabilities for managing the syntax of a domain as part of its data structure (i.e., the number of characters in a field and the number of usable functions are usually small);
- explicit relationships between data classes cannot be represented (i.e., those not based on the semantic data model); [Yao85]
- database interactions demand precise, mathematical query formulation;
- many of the data objects used in the application domain do not exist explicitly in the database;

- image data (e.g., multi-dimensional point data, polygons, raster, etc.) cannot be properly stored, indexed, or retrieved in relational systems;
- elements are either in or out classes (i.e., attributes), despite the fact that humans use fuzzy classes (i.e., sets) where one element may intuitively be more in a class than in another; [Rosch1973]
- interfaces that use rigid hierarchies are not supported by cognitive evidence that humans use only one standard hierarchy; [Rips1973] [Rosch1973]

3. OVERALL DESIGN

3.1. CONCEPT

The UI is a system that will serve as an intermediary between a database and a user who has little or no knowledge of the database architecture, data content, query language, or analysis procedures. Such an interface would allow the user to operate a database (i.e., identify and select data) in the context of a user's particular knowledge domain by using media such as expert advice, natural language (English), icons, pictures, and images. The advantages of such a concept are that :

- It can facilitate an understanding of the database by specifying the contents and meanings (i.e., relationships) of the data as well as the relation between objects (i.e., actual database objects or cluster of objects) within the data structure; [Campbell87]
- it can aid in understanding and memory recall (i.e., user education) by correctly associating language, pictures, etc., in the presentation to the user; [Bransford1972]
- It will support approximate reasoning to infer conclusions that are not explicitly stated by the user, such as imprecise or fuzzy queries which can be stated without mentioning data names or operations;
- It can provide the casual user with a logical representation of the database architecture, the stored data, and the analysis procedures; [Campbell87]
- It can serve as a foundation for database schema modification through the examination of user queries and through explanation-based learning systems; [Lanka85]
- It can link different worlds (e.g., database, graphics, image, etc.) over a network by reducing user goals into a plan based on processor/network loads, the type of application, and specific syntax.

In our view, the interface process can be broken into two separate phases. The first process is to determine, based on the user's utterances, his motivation or data access goal. In other words, this first step is a simple process of abduction of the user's plan (i.e., motivation analysis) using his terminology. [Charniak85] Once the user-goal is found, the second process involves reducing the goal to its primitive operational steps. These steps could be as simple as operating system commands or as complicated as a natural language query (at least complicated in the generative procedure).

One implicit assumption is that all the user's goals are finite, or at least capturable from a database or domain expert. We will assert (without proof) that the majority of the user community's goals can be partitioned according to a specific discipline (e.g., a project manager, geologist, or database administrator). Although not necessary, these partitions, called application views in our argot, correspond roughly to a forest of semantic networks which contain both tasks, domain-specific knowledge, and user-profile knowledge. For example, user-profile knowledge is used to determine where to start questioning in application view, as well as which terminology to use. Once in a view, if a misclassification occurs, either a re-classification can be done or the user can be directed to a default view, called the architecture view. This view is intended to be a shallow, but broad, view of the system's uses. For example, in the database world, this may involve only a type hierarchy of all the attributes.

3.2. PROJECT OVERVIEW

The database selected, the Crustal Dynamics Data Information System (DIS), is used to support NASA's Crustal Dynamics Project. The DIS was selected because of its consistent architecture, the IUI developers' close working relationship with the DIS developers, and the problems of large databases exhibited in the DIS. (The DIS contains over 200 relations).

In general, the crustal project is studying a) the motions of the earth's plates b) regional deformation of important seismic areas (e.g., San Andreas fault zone) c) polar motion and earth rotation and d) the relative stability of the plates. To calculate these movements, two techniques are used: satellite laser ranging (SLR) and very long baseline interferometry (VLBI). In the SLR technique, a pulse of laser light is transmitted from a ground station (either fixed or mobile) to a retroreflector in Earth orbit. By measuring the round-trip travel time for the pulse and knowing accurately the orbital dynamics of the satellite, it is possible to locate the station on the surface of the Earth in a Earth-centered coordinate system. Observations from many stations permit the calculation of baselines (i.e., baseline, chord, and geodesic measurements) between these stations. By making similar observations over extended periods of time, it is possible to determine the change in such a baseline length. Even though the goal of determining distances is the same, the VLBI technique is different in that it uses the correlation of radio signals received from extremely distant radio sources (quasars) to determine baselines.

The original crustal IUI prototype was done on an IBM PC/AT and a Vax 11/780. While this system aided in rapid prototyping and knowledge engineering, it ultimately failed due to the inherent problems with the PC. [Short87] With the acquisition of a Sun 3/260, the next step was transfer the knowledge garnered to a multi-tasking environment that contained better development tools.

Given the limited resources in personnel, we have chosen as a philosophy to incorporate as many off-the-shelf development packages into the interface as possible in the desire for eventual standards. Being integrationists causes several problems to arise. For example, our experience has been that many of the vendors have not encountered these esoteric problems and, therefore, are of little help in the integration phase. Second, by integrating several levels of software (e.g., Fortran to C to LISP), each level presents its own set of errors, allowing us no assumptions of whether the system works anywhere. Lastly, because many of the packages are black boxes, much trial-and-error programming is required to ferret out all possible user interactions. We, therefore, intended this version to be a test of the possible problems associated with the integration of multifarious commercial packages.

4. EXAMPLE CONSULTATION

Generally, the intent of the system is to have the user, if he/she feels comfortable, generate his/her own query to a commercial natural language front-end. If the result is unintelligible or the user does not know what to specifically ask, then he/she is directed to use the database advisor expert system. This advisor will ask the user a series of questions and use the answers to formulate an English query, which is then sent to the natural language processor for the data results.

Before getting into the physical architecture, we will show an example session of how a user might get information about the crustal plates. Suppose the user wished to know the stability of the South American plate. Under the current DIS (i.e., without the IUI), he/she could issue the following SQL query:

```
SELECT    baseline80_slrgsfc.f_station,baseline80_slrgsfc.s_station,
          baseline80_slrgsfc.baseline,  baseline83_slrgsfc.f_station,
          baseline83_slrgsfc.s_station, baseline83_slrgsfc.baseline
FROM      baseline80_slrgsfc, baseline83_slrgsfc, sites
WHERE     baseline80_slrgsfc.f_station =baseline83_slrgsfc.f_station    and
          baseline83_slrgsfc.s_station = baseline83_slrgsfc.s_station    and
          (baseline80_slrgsfc.f_station = sites.station                  or
           baseline80_slrgsfc.s_station = sites.station)                  and
```

```
sites.plate = 'south american';
```

At first glance, a novice user might be inclined to ask the natural language front-end the question:

"What is the stability of the South American plate?".

From the above SQL query, it should be intuitively obvious that this query would not satisfy the constraints of year (80, 83), distance type (baseline, chord, geodesic), etc. In fact, the natural language front-end would actually send back

<u>STABILITY</u>	<u>STATION</u>	<u>SITE_NO</u>	<u>PLATE</u>
good	7108	31	south american
fair	7500	22	south american
poor	7800	22	south american
.	.	.	.

as an answer, meaning there existed a stability field accounting for the qualitative stability of each station on the plate. The solution to this problem would be to consult the expert system as in Figure 1. One should notice the semantic difference in the query generated by the expert system ("For the South American plate, show me the slr baselines for 1983 and the slr baselines for 1980 using the same stations in 1983 and 1980") and the one asked by the novice user above.

5. COMPONENTS OF SYSTEM

In addition to the integration goal, a major intent of the system was to mimic on a small scale the concept of a local scientific workstation connected to a larger, central system. This means that the tools available to the scientist are in general the same for the larger system, except the nature and amount of data vary. The point of this is to allow the scientist to capture data into his system so that he can modify his data as well as structure. Customization of the workstation then means changing the local knowledge base (that is a subset of the central), the database schema (e.g., into views), and the data per se to fit the scientists needs. For example, an explanation-based learning system (see paper in this conference on "The Advice Taker/Inquirer, a System for High-Level Acquisition of Expert Knowledge") could be used at both the central and local level to modify the knowledge-base as well as the database schema (see section 6.2). For the data capture, a simple planner could be used to, given the different data locations and system loads, capture and format the data into a readable form at the local level. Value-added knowledge and data results can then be reshipped to the central source if the scientist and project team so desires.

In practice, only several of these components were integrated (i.e., in time for the paper) into a demonstrable system, as some of the components are still under development (e.g., the explanation-based learning system, AT/I). See Fig. 2 for an overall systems design of the intended system as well as the actual.

Physically, the local and central system reside on the Sun 3/260 (Sun Unix BSD 4.2) and the VAX 11/780 (VMS 4.7) respectively. The Sun 3/260 contains the Automated Reasoning Tool (ART) by Inference Inc., the The NLI DataTalker and the NLI Connector by Natural Language Inc., the Sybase DBMS by Sybase Inc., Figaro by TEMPLATE Graphics Inc., and LISP,C, and Fortran as languages. Also, the software on the VAX contains Figaro, THEMIS natural language front-end, and the ORACLE DBMS.

5.1. DATABASE MANAGEMENT SYSTEM

ORIGINAL PAGE IS
OF POOR QUALITY

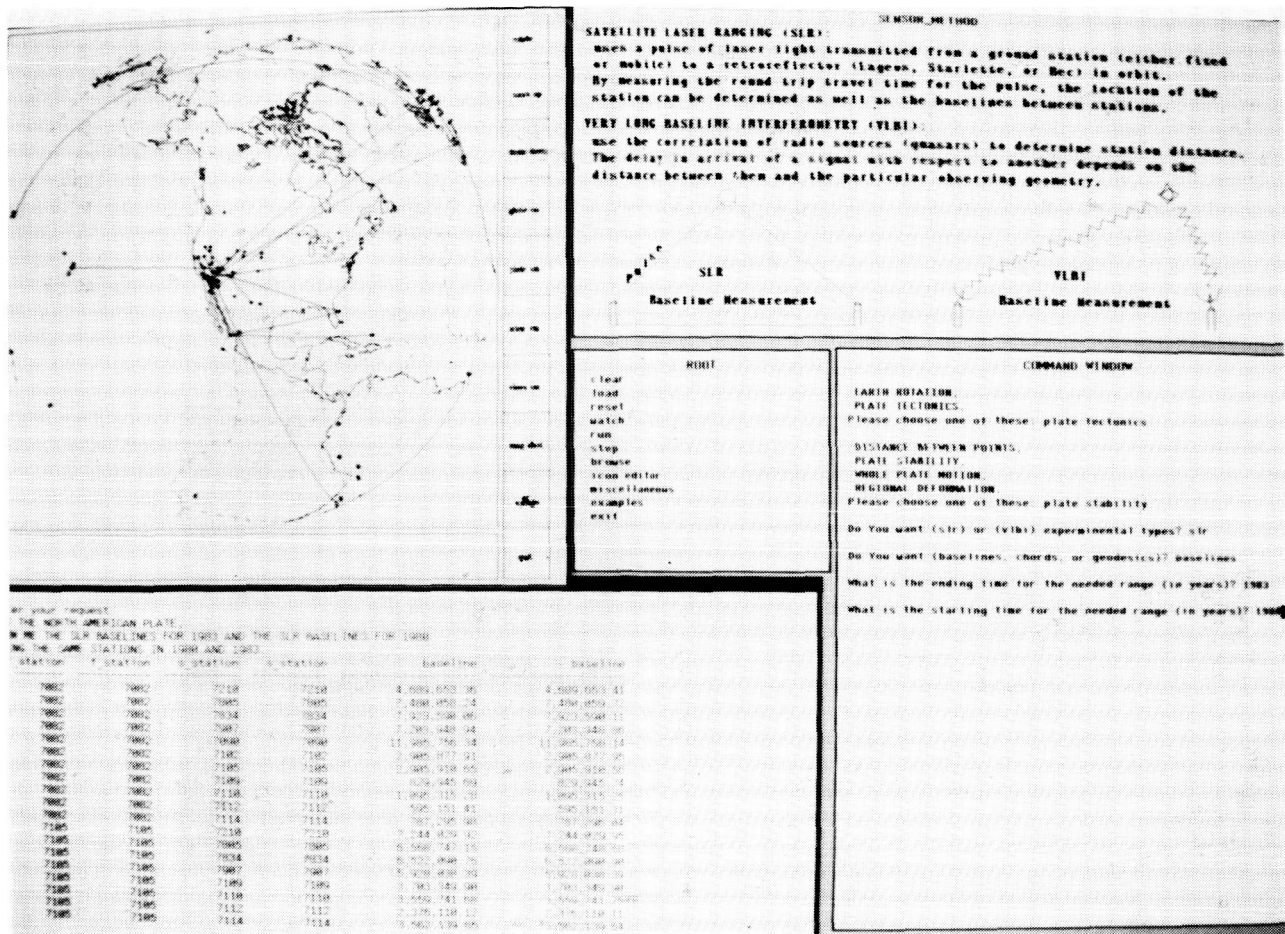


Fig. 1 EXAMPLE CONSULTATION

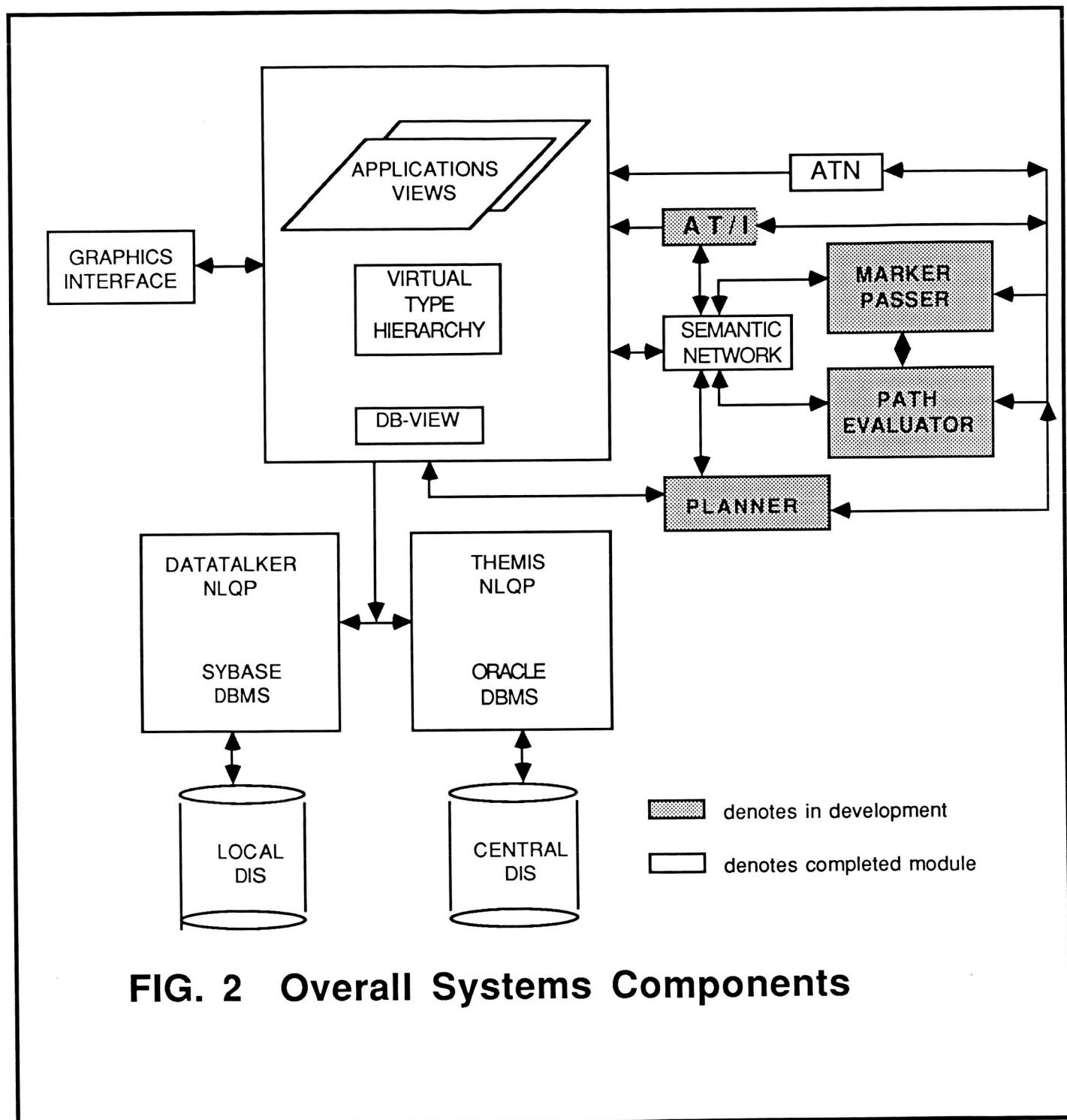


FIG. 2 Overall Systems Components

As aforementioned, two DBMSs were used to 1) test the effects of using the different DBMSs with two natural language front-ends and 2) mimic the natural design of having a central DBMS with a subset DBMS residing on the scientific workstation.

5.2. NATURAL LANGUAGE QUERY PROCESSOR

More for historical reasons on the project, the THEMIS natural language front-end was used to serve the central DIS while DataTalker resided on the SUN as the local database server.

In THEMIS, the parser consisted of an ATN-based (Augmented Transition Network) system, which contained limited semantic capabilities. First, THEMIS handled intersentential referential ambiguity by referring to the last dangling referent in the discourse. [Charniak85] Second, because of this limited amount of context determination and the lack of knowledge about the user, a precise level of detail was required in the English query. Finally, once the data result was returned to the user, no attempt was made to evaluate the intensional accuracy of the query. [Mays80]

DataTalker, on the other hand, was a marked improvement over THEMIS because of the addition of a simple inference engine. This allowed the user to ask more general queries, in addition to gaining better explanations about intensional errors. [Mays80] Furthermore, DataTalker's graphics facility allowed, for the first time, some analysis of the data results by providing 2D graphs (i.e., pie charts, bar graphs, x-y plots, etc.). Unfortunately, the lack of substantial discourse capabilities and the fact that not all queries were parsable still implied the need for a database advisor expert system.

5.3. EXPERT SYSTEM

Written in ART, the expert system database-advisor/controller was sectioned into two domain-specific views of the DIS: the single applications view for plate tectonics and the architecture view. Basically, the topology of each view was organized into an and/or graph using ART's viewpoint mechanism. At the initial time of consultation this forward-chaining rule,

```
(Defrule establish-application-view
"generate the entire application view"
  (declare (salience *constraint-salience*)) ;; highest priority rule if
  (current-view application-view)           ;; the view is the application view.
=>
  (sprout (assert (type world)
    (user-desires world) ;; starts the consultation.
    (old-type-register world) ;; points to parent type
    (sprout (assert (type plate-tectonics) ;; child of world type
      .
      .
      .
      )))),
```

would establish the initial tree using the "sprout" command, which spawns off a viewpoint world. In general, an ART viewpoint is a "...means of segregating data [assertions] into separate modes of the situation that an application is considering" in much the same way a frame in a situation calculus is organized.[ART87].

In this viewpoint case, the "type" predicate (relation) is used to describe the situation in which the user is working. In other words, if the user feels that this type is detailed enough, he can force a series of goal-directed rules of the form

```
(defrule Line-type ""
  (goal (line-type ?type)) ;; if there is a goal to find the type of line
  (not (line-type ?type)) ;; and that fact does not exist already.
```

```

<other pre-conditions>
=>
  (printout t t "Do you want (baselines, chords, or Geodesics)")
  (assert (line-type =(nlp-read))          ;; read from natural language parser.
    (remove-window "measurement"))        ;; remove the help window

```

to create the necessary deep constituent structure needed in generating the specific English database query.

To traverse this viewpoint structure, the following rules

```

(Defrule set-child-nodes
"make a list of the lower viewpts for traverse-tree to pick up"

  (declare (salience 10))          ;; declare the priority of this rule on the agenda.
  (viewpoint ?vp                    ;; find a viewpt. with the same ?type variable
    (user-desires ?type)           ;; for the predicates, user-desires and type
    (type ?type))

=>
  (at ?vp (assert (lower-childs =(lower-contexts ?vp))))

(Defrule traverse-tree
"get the user's choice of the lower child viewpts"

  (declare (salience 9))
  (viewpoint ?vp
    ?x<-(user-desires ?type)
    ?y<-(type ?type)
    ?z<-(lower-childs ?children))
  (for ?n in$ ?children do          ;; print out lower types.
    (viewpoint ?n (type ?type-2&~?type)) ;; ?type-2 and not ?type.
    =>
      (printout t t (format nil "~{ ~S~}," (list$ ?type-2))))
  ?A<-(old-type-register ?old)      ;; get old parent from a
                                   ;; register predicate.

=>
  (printout t t "Please choose one of these: ")
  (at ?vp (retract ?x ?y ?z)        ;; remove these facts from
    (assert (user-desires =(seq$ (nlp-read)))) ;; viewpoint ?vp.
    ;; nlp-read does I/O, so to
    ;; assert a new user-
    ;; desires.

    (retract ?A)
    (assert (old-type-register ?type)))

```

basically, produce the query to the user such an example

```

PLATE TECTONICS,
EARTH ROTATION,
Please choose one of these:

```

Of particular note, the user need not respond using these answers, but could in fact have responded in English. This would be done for the following reasons:

- 1) the user realizes that he doesn't know his specific needs;
- 2) the user has led the system down the wrong reasoning path by entering inappropriate responses;
- 3) the user realizes that he doesn't want what the system is providing;
- 4) the user has several different responsibilities, such as a project manager who is also an engineer;
- 5) the user wishes to ask questions about the knowledge base's structure;
- 6) the user wishes to control directly a world (e.g., graphics) via natural language (e.g., "rotate the globe right").

In cases 1-4, the sense of the user's response (or question) would be used in determining which point in the architecture view the systems should branch to and what parameters need not be filled in the database query (e.g., line-type) [Short87]. To parse this English sentence, a standard ATN was developed for the syntactic parse. Specifically, an ATN compiler was coded to take statements such as this

```
;;; S: np -> s-vp
      (defstate s
        (parse np t s-vp (setr subj *value*)))
      ;; starting state
      ;; arg-1--arc type
      ;; arg-2-- test
      ;; arg-3--destination state
      ;; arg-4--side-effect action
```

in order to produce a parse tree, in which the semantics could be done in ART and LISP. The compiler, of course, could handle transformations (e.g., aux-inversion, you-insertion, etc.), subject-verb agreements, etc. and backtracked chronologically (at least at this point--see sect. 6.4). [Charniak85]

Of course, if the branching point could not be determined from the ATN, then the user would be sent to the top of the first component of the architecture view. This first component, called the **virtual type hierarchy**, consists of a sample of query templates that would apply to all of the database domains and could be determined from the database administrator (DBA). In other words, by using the list of "canned queries" which the DBA usually keeps online, the DBA can give an explanation of the highest requested queries. These canned queries are then aggregated and generalized into a standard type hierarchy. [Sowa84] [Smith77]

The data structure for the virtual type hierarchy is the same as the applications view. Once it has been determined that the user is uncertain, the entire virtual type hierarchy viewpoint structure is then spawned off the last viewpoint traversed. Although the entire viewpoint structure would be expanded, the ATN would force the branching point by asserting, for example, (user-desires site-location). This would trigger "generate-child-nodes" in the site-location viewpoint and not in the top level viewpoint, "virtual-world." By spawning the entire structure, the system can save the effort of expansion during the next iteration of the consultation.

Because the virtual type hierarchy does not cover all the possible queries a user can ask, the lowest level view, the **database view** (db-view), is entered if the user enters "uncertain" while he/she is in the virtual type hierarchy. In general, the db-view is a bottom-up schema hierarchy starting with the following frame:

(defschema attribute-frame "a generic frame for a database attribute"

```

  (attr-type [virtual, real, view] ) ;; virtual: means exists in the know. base but not in
                                     ;; the central or local db; note data can
                                     ;; exist as has-elements for this frame
                                     ;; real : means exists in the central db
                                     ;; view : means is a view of the central db, but
                                     ;; located in the local db
  [(syn ...[attribute-synonyms])] ;; synonyms of the attribute name
  [(pk ...[primary-key])]         ;; points to the attribute's primary key
  [(value-types ...[attri-values])] ;; for attributes that have class values (row-wise)
  [(icon-picture ...[icon-referent]);] points to the schema that has the picture of the
                                     ;; object (i.e., like farkle units)
  [(descr ...[(description-list)]] ;; contains a list that can be used to describe the
                                     ;; object for help purposes
  [(help-window ...[window-referent]);] ;; help window for complex objects.

```

At the lowest level *n*, each database attribute is an instance of the attribute frame and forms the leaf of the tree. Instead of inheriting all the information from the root, the attribute's frame passes its relations (slots) to its parents. An actual relation in the database is, therefore, the frame at level *n*-1. The lower levels (< *n*-1) are then super relations based on aggregation and generalization. [Smith77]

While the kernel of this db-view is an "isa" hierarchy, knowledge about the domain, called virtual objects, and procedural knowledge (generic operations on the data) can be appended through accretion to this semantic network. This allows rules in any of the views to access general information about the domain. Also, using for example,

```

(defaction baseline-change ((l-1 length) (l-2 length)) ()
  (do ((lst1 (list$ (get-schema-value 'l-1 'baseline)) (cdr lst1)) ;; get baselines
      (lst2 (list$ (get-schema-value 'l-2 'baseline)) (cdr lst2))
      (result nil))
    ((or (null lst1) (null lst2)) (modify-schema-value 'baseline
                                                         'delta-baseline
                                                         (seq$ (reverse result))))
    (setq result (cons (- (car lst1) (car lst2)) result))) ;; subtract each
                                                         ;; baseline value

```

as a procedural object, the system can invoke the message passing from object-oriented programming to determine, in this case, the change in a baseline.

If needed, this db-view isa hierarchy can be traversed in a manner similar to the other views. This requires that the schema data structure be converted to a viewpoint hierarchy. In our case, instead of impractically converting the entire semantic network, only one level was added at a time using, for example, the following rule:

```

(defrule generate-viewpoint-level-t1 "type1- expand on instance-of relation"
  (declare (salience *constraint-salience*))           ;; highest priority
  (view-destination ?type & ~architecture-view & ~application-view) ;; ?type points to
                                                         ;; parent.
  (test (equal *predicate-expansion-type* 'instance-of)) ;; *var* determines
                                                         ;; what to expand on.
  (viewpoint @?vp                                       ;; find a viewpt with
                                                         ;; ?type-2 at the
                                                         ;; highest level

    (type ?type-2)
    (user-desires ?type-2))
  (schema ?type                                         ;; catch all ?type that
                                                         ;; exist on
                                                         ;; instance-of in
                                                         ;; db-view.
    (instance-of ?real-type))

=>
  (at ?vp (sprout (assert (type ?real-type))))).

```

In this rule, the view-destination relation's variable argument, ?type, is used to expand levels if the schema exists. Also, the root for ?type, as usual, can be determined by default or from the ATN. Lastly, the level generation can be terminated when either the level n-1 (the db relation level) is reached or the user types "all." In the latter case, all the attributes below would be used in the construction of the query(s).

In addition to serving as low-level query construction process, the db-view also suits as a foundation for schema modification and data ingest into the knowledge base. For example, If a user wished to create/modify his own schema, the system would modify this portion of the system to allow inheritance of known information. With this modification, the system could generate the appropriate SQL (or English in the Datatalker case) to create a view or new relation in the database (note: this has not been done, yet). In general, these rules for this relation creation would be triggered if the arguments to the attribute's slot had reached a point where storing data in the knowledge base as arguments was inefficient. This threshold would be a function of the total data stored in the knowledge base.

In conclusion, the expert system and natural language components can be viewed as a process of going from general to specific, as required by the user. At the most general level, the applications view serves as a "guess" at what the user desires. While it contains more knowledge about the domain, it produces the fewest number of queries. The next level, the virtual type hierarchy, represents an intermediate level between expertise and specifics from the database schema (or data dictionary) and is appropriately derived from the DBA. The last expert system level, the database view, contains all the specific information for the system to generate the most queries and modify its knowledge about the network's or system's domain. Lastly, the lowest level is the natural language component because of its lack of any assistance in query formulation and its requirement for a high level of specificity in the generation of complex queries.

5.4. GRAPHICS INTERFACE

From the scientists point of view, the collection of measurements is used to build a spatial and temporal model of the movement of the crustal plates. Although the measurements themselves can be maintained in a relational table, the data must be recast in the visual domain to assist spatial problem solving. For instance, many of the objects in the relational tables, such as the site of the measurement, are identified by a unique but arbitrary key. Having to manually look up the site id for the site in Venezuela assumes the user has a printed table handy, or is willing to create more queries to find that piece of information.

Although the site location may also be unique, there is a hidden difficulty in using relational calculus to find a spatial object -- unless the location is specified with exactly the same number of digits as is stored in the

relational table, the relational search will fail, simply because too many or too few digits of accuracy are specified. Spatial concepts (called topological relations) such as **near**, **contained-within**, **closest-neighbor**, **overlap**, and the like, are poorly or not at all representable as relational queries. [Charniak85] Computer graphics, thus, has a two-fold purpose: to visualize information to aid spatial problem solving and to assist in identification and searching of spatial objects. By permitting the usage of picking or pointing devices in interactive graphics, alternative spatial search techniques are implicitly added.

From a spatial perspective, the relational database is useful as a robust data ledger that needs maintenance and updating but is intrinsically inadequate for searching for data on the ledger when some fuzziness or non-relational relationship is implied within the data.

The spatial objects in the crustal dynamics relational tables are measurement sites, with measurement baselines between various sites taken over several years. Only selected measurements are made. In the first prototype on the PC/AT, sites were plotted as small icons on a two dimensional projection of the earth's coastlines, the Mercator projection, along with plate boundaries. Note that neither the coastline data nor the plate boundary data is stored in the relational database. This bit of world knowledge comes from outside of the database, yet, is essential for a meaningful spatial representation of the information within the relational database. That is, the coastlines and plate boundaries are necessary for conveying a spatial reference frame of the data. [Short87] The spatial objects thus visualized were sites (two types), in the context of coasts and plates. The graphics interface was built using the Graphics Kernel Standard, or GKS.

In the second generation prototype on the Sun 3/260, the Mercator projection was abandoned -- some baselines crossed the international date line, and could not be well represented using a flat projection. A three dimensional graphics package based on the Programmers Hierarchical Graphics Standard, or PHIGS, was used to build a three dimensional model -- a ball -- of the earth. Coastline data and plate boundaries were converted from latitude/longitude to (x,y,z) on a sphere. A menu was added with buttons which permitted the sphere to be rotated and zoomed. So, any point on the earth could be made visible, although the final 3D to 2D viewing transformation would only show the visible side of the sphere. Rather than a static map, the earth was a three dimensional object that could be manipulated as needed. At low resolution, site icons were represented as little more than triangles or squares, depending on site type. At high resolution, the site icons were represented as simple, three dimensional objects (although still not to scale).

The transformation from a two dimensional representation to a three dimensional object gave a more accurate representation of the spatial problem. However, this was not without a price: both the viewing and searching strategies became more complex. For example, determining if a point is inside or outside a polygon in two dimensions is straightforward; that is, choose a point outside the polygon, such as a point at infinity, and count the number of times the line between the outside point and the unknown point crosses the polygon. If the count is odd, the point is outside; if the count is even, the point is inside. The special case of a tangent line can be checked. On a sphere, it is not possible to arbitrarily choose a known outside point the polygon -- infinitely or arbitrarily large points do not lie on the sphere, and are thus invalid choices. In general, most simple plane geometry algorithms had to be redesigned when applied to a three dimensional sphere.

Another problem is the choice of coordinate systems. The latitude/ longitude coordinate system is a poor choice -- the international dateline becomes a special case in almost every algorithm. The Cartesian (x,y,z) coordinate system is continuous, and so avoids a lot of special case logic, but is not as compact as a two coordinate system. In fact, the spatial structures built with the (x,y,z) coordinate representation are one third larger. But for most applications, including this one, it is less expensive overall to use more memory and less special case programming. As another problem, the viewing transformation was more complex, requiring cross products of three dimensional vectors to determine the most suitable viewing transformation. However, the math is only slightly more expensive in terms of computation, and the reward was correct clipping of baselines that went over the horizon.

6. FUTURE DIRECTIONS

Rather than interfacing to a single database world, an information system could be managed by an expert system that controls several databases, selecting the most appropriate data model, based on the type of information and the goals of the user. The expert system would be responsible for the user interface--

that is, understanding the goals of the user and translating those goals into the query language of the appropriate data model or world.

Of course, these worlds would not coexist on a single machine but would be spread over a network of machines dedicated to particular applications (e.g., the Massively Parallel Processor is good for imaging and graphics). As the network size increases, the ability of the user to conceptualize all the possible domains and alternatives will decrease, forcing more control by the system (see 6.3). Thus, the following section illustrates a few of the needed components to extend the interface into an Advanced Information Management System.

6.1. GIS FOR SPATIAL QUERIES

For some applications, usage of systems or techniques that have evolved from geographical information systems (GIS) could be used to provide representation for spatial types and a query language for spatial searching. First, for extended raster based objects, a quad tree GIS system may be best. Second, for point-like data, a k dimensional tree may be best. Third, for some applications, a computer assisted drafting tool based on a vector representation may be used. Finally, for sensor-based observations of the earth, a GIS may be the most appropriate tool.

6.2. KNOWLEDGE ACQUISITION

With the evolution of these computing environments to the network and the inability to access qualified knowledge engineers in languages such as ART, it will become increasingly necessary to incorporate explanation-based learning paradigms that capture the expert knowledge at the workstation level. For this reason, we (IDM lab) are currently developing the Advice Taker/Inquirer [Cromp88]. In context of this system, such a tool could aid in the development of applications views, which would be shared by scientists on the network community. For example, if a user desired to access a database belonging to another organization, he would ingest the appropriate application views before querying the actual database. An architecture view would then serve as a *heuristic view* for "guessing" where the relevant knowledge on the network resides.

6.3. PLANNING/SCHEDULING

A natural problem arises when using the heuristic view to guess which of the environments is best to use. For example, suppose we knew that ART existed on three machines with the desired application view existing on machine 1. Also, suppose that machine 1 is saturated. We, thus, have an alternative of either using machine 1 to solve our query problem or transporting the knowledge to machine 2 or 3. Now, we could use standard approaches to calculate the best alternative, but over a large network this would be impractical for both the system and user. For this reason, we are developing a planner (Noah-class) that would reduce a user's goal into a plan over the network environment and would take into account the applications characteristics (e.g., some applications run better on some machines) and the time dependency problems.

6.4. MARKER PASSING

As the semantic networks of these systems increase, problems with unification will force the need for subsetting the knowledge base based on the context. To solve this problem, we are exploring the use of marker passing (spreading activation) to aid both in the directing of the ATN's parses and the choosing of plans for solving user goals. [Charniak86] [Hendler88]

7. CONCLUSION ON INTELLIGENT DATA MANAGEMENT

As aforementioned, the major goal of this development phase was to determine the integration problems and the required techniques (e.g., planning) for a complete Advanced Information System. In addition to achieving this, several specific conclusions were reached. For example, the process of creating a hierarchy "on-the-fly" became important as a way of presenting the same information differently (i.e., relativism) and establishing a structure for eventual user modelling. [Short87] Also, treating views as entities allows for the sharing of scientific knowledge as well as scientific data. That is, if a scientist wants to use, verify, and extend another's work, he could determine the correct applications view, ingest it, and use that to understand the intent of the data analysis. Unfortunately, as network community sizes increase, the scientist's ability to browse all the available tools and knowledge will diminish. Thus, it is our belief that these systems which automatically locate information based on scientist's requirements will be crucial in information operations of any institution, regardless of its scale.

8. ACKNOWLEDGEMENTS

The authors would like to thank the following people for their valuable help during the development stages: Seth Allen/Science Applications Research (SAR), Bill Campbell/Code 634, Bob Crompt/SAR, Larry Roelofs/Computer Technology Associates, Lloyd Treinish/Code 634, Sam Mathews/SAR, Carey Noll/Code 634, and Henry Linder/Code 634.

9. BIBLIOGRAPHY

Art Reference Manual Ver. 3.0, Inference Corp., Copyright 1987.

Bic, L. and Gilbert, J., "Learning from AI: New Trends in Database Technology," IEEE Computer, Mar 1986, p. 44-54.

Bransford, J. and Johnson, M., "Contextual prerequisites for understanding: Some investigations of comprehension and recall," Journal of Verbal Learning and Verbal Behavior, 1972, 61, p. 717-726.

Campbell, W., Short, N. Jr., Roelofs, L., and Wattawa, S., "The Intelligent User Interface For NASA's Advanced Information Management Systems," Third Conference on Artificial Intelligence for Space Applications, Nov. 1987.

Campbell, W.J., Roelofs, L., and Short, N. Jr., "The Development of a Prototype Intelligent User Interface system for NASA's Scientific Database Systems," NASA TM-87821, Apr 1987.

Charniak, E., "A Neat Theory of Marker Passing," Proceedings from the Fifth National Conference on Artificial Intelligence, Aug 1986, p. 584-88.

Charniak, E., and McDermott, D., Introduction to Artificial Intelligence, Addison-Wesely Publishing, Co., copyright 1985.

Clayton, B., ART Programming Tutorial, Vol. I-III, Inference Corp., copyright 1986.

Crompt, R., "The Advice Taker/Inquirer, a System for High-Level Acquisition of Expert Knowledge," Proceedings from Goddard AI Conference 1988.

Report to the Chairman, Committee on Science, Space and Technology, Space Operations, NASA's Use of Information Technology, GAO/IMTEC-87-2-, April 1987, p. 46-50.

Hendler, J., Integrating Marker-Passing and Problem-Solving: A Spreading Activation Approach to Improved Choice in Planning, Lawrence Erlbaum Associates, Inc., copyright 1988.

Mays, E., "Failures in Natural Language Systems: Applications to Database Query Systems," Proc. First Natl. conference on Artificial Intelligence, 1980, p. 327-30.

Kneale, D., "What Becomes of Data Sent Back From Space? Not a Lot, as Rule," Wall Street Journal, Jan. 11, 1988.

Lanka, S., "Automatically Inferring Database Schemas," Proceedings of the Ninth International Joint Conference on Artificial Intelligence, Aug. 1985, p.647-49.

Rips, L., Shoben, E., and Smith, E., "Semantic distance and the verification of semantic relations," Journal of Verbal Learning and Verbal Behavior, 1973, 12, 1-20.

Rosch, E.H., "Natural categories," Cognitive Psychology, 1973, 4, p. 328-350.

Short, N. Jr., et.al. "The Crustal Dynamics Intelligent User Interface", NASA TM-100693, Oct 1987.

Smith, J.M. and Smith, C.P., "Database Abstractions: Aggregation and Generation," ACM Transactions on Database Systems, Vol. 2, No.2, June 1977, pges. 105-133.

Sowa, J., Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley Publishing, Co, copyright 1984.

Treinish, L., Campbell, W.J., Roelofs, W.J., "Graphical Manipulation, Management and Display of Hierarchical Data Structures," Fourth Conference of the Template User Network (TUN), Mar. 1987.

Yao, S. B. (ed.), Principles of Database Design, Prentice-Hall, Inc., copyright 1985.

SPACELAB DATA PROCESSING FACILITY

(SLDPF)

QUALITY ASSURANCE (QA) / DATA ACCOUNTING (DA)

EXPERT SYSTEMS:

TRANSITION FROM PROTOTYPES TO OPERATIONAL SYSTEMS

ABSTRACT

The SLDPF is responsible for the capture, quality monitoring, processing, accounting, and shipment of Spacelab and/or Attached Shuttle Payload (ASP) telemetry data to various user facilities. Expert systems will aid in the performance of the quality assurance and data accounting functions of the two SLDPF functional elements: the Spacelab Input Processing System (SIPS) and the Spacelab Output Processing System (SOPS). Prototypes were developed for each as independent efforts. The SIPS Knowledge System Prototype (KSP) used the commercial shell OPS5+ on an IBM PC/AT; the SOPS Expert System Prototype used the expert system shell CLIPS implemented on a Macintosh personal computer. Both prototypes emulate the duties of the respective QA/DA analysts based upon analyst input and predetermined mission criteria parameters, and recommend instructions and decisions governing the reprocessing, release, or holding for further analysis of data. These prototypes demonstrated feasibility and high potential for operational systems. Increase in productivity, decrease of tedium, consistency, concise historical records, and a training tool for new analysts were the principal advantages. An operational configuration, taking advantage of the SLDPF network capabilities, is under development with the expert systems being installed on SUN Workstations. This new configuration in conjunction with the potential of the expert systems will enhance the efficiency, in both time and quality, of the SLDPF's release of Spacelab/ASP data products.

Lisa Basile/Code 564.2
NASA/Goddard Space Flight Center
Greenbelt, MD 20771

May 1988

TABLE OF CONTENTS

1. INTRODUCTION
2. DESCRIPTION OF PROTOTYPES
 - 2.1 SIPS KSP
 - 2.2 SOPS ES PROTOTYPE
3. SLDPF RESPONSE TO PROTOTYPES
 - 3.1 REALIZED IMPROVEMENTS
 - 3.2 ADDITIONAL DESIRED ENHANCEMENTS
4. OPERATIONAL EXPERT SYSTEMS
 - 4.1 SYSTEM CONFIGURATION
 - 4.2 THE SIPS ES DESIGN
 - 4.3 THE SOPS ES DESIGN
5. ES MOTIVATED IDEAS WITHIN SLDPF
6. SUMMARY

1. INTRODUCTION

In early 1986, the SLDPF operations area was considered a prime candidate for expert systems applications within the GSFC Information Processing Division (IPD). In particular, the QA/DA analyst functions of both the SIPS and the SOPS exhibited the potential to expedite operations with the aid of expert systems; extremely large volumes of data from one mission to another and the short turnaround requirement for delivery to users makes the QA/DA task both demanding and tedious. The objective of the expert systems is to assist analysts by making decisions and suggesting operational procedures based on given data quality information and mission criteria.

The benefits presented by the expert system prototypes, developed for both the SIPS and the SOPS, convinced the IPD personnel that operational systems would be advantageous to the performance of the SLDPF. Project approval was granted in May 1987. Since then work has been progressing toward delivery dates in time to support the next scheduled SLDPF mission.

2. DESCRIPTION OF PROTOTYPES

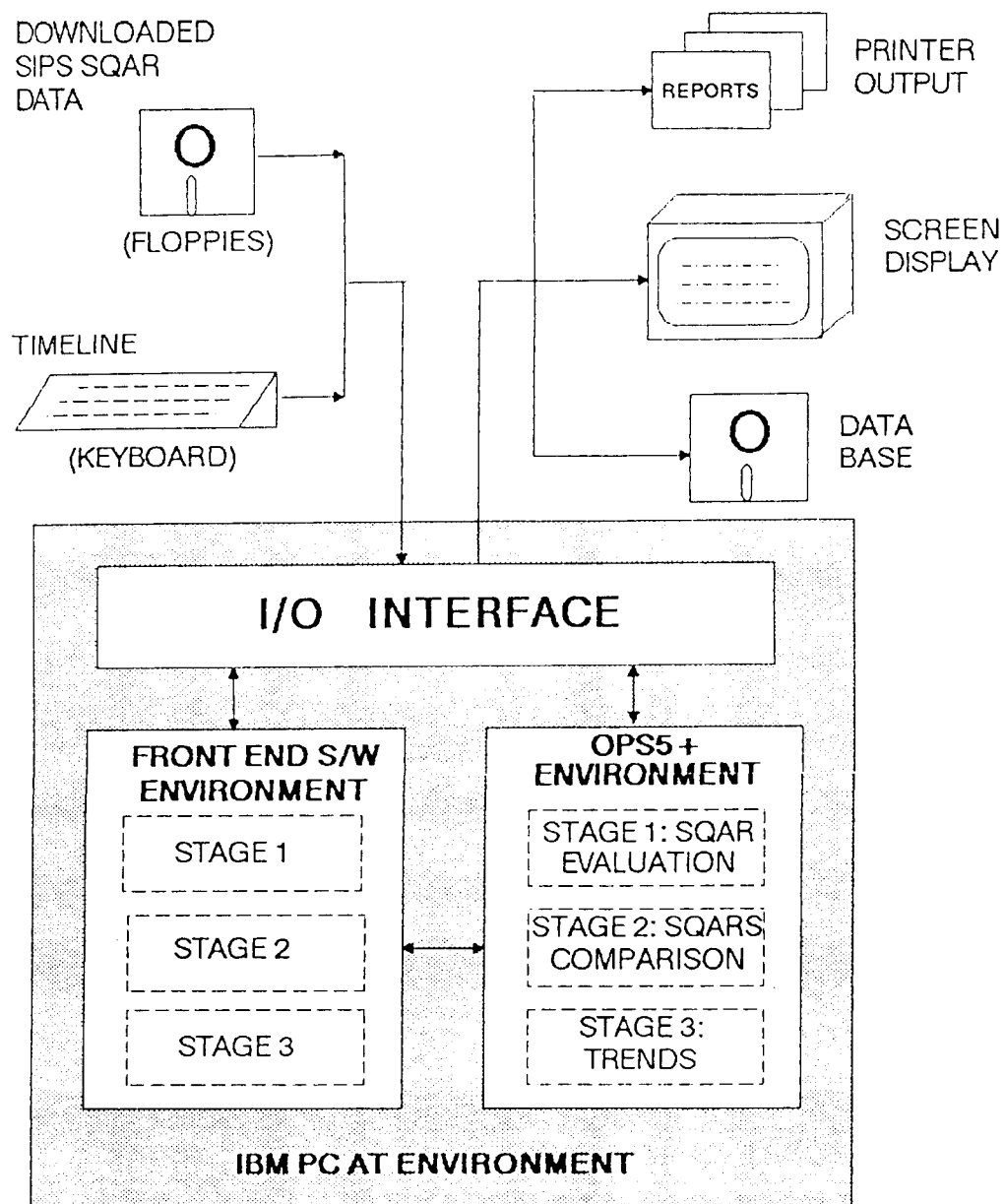
The strategy adopted was to develop the prototypes on personal computers using commercial expert system shells; then, build the QA/DA knowledge bases within the shells.

2.1 THE SIPS KSP

The SIPS KSP was designed to emulate the performance of experienced SIPS QA/DA analysts in the evaluation of Spacelab data quality and accounting information, a function previously performed through the examination of data quality and accounting reports. The KSP was developed using the OPS5+ Development System, a rule-based expert system shell; a MicroSoft "C" compiler; and Information Builders, Inc. PC/FOCUS software installed on an IBM PC/AT. The scope of the initial effort was restricted due to the extensiveness of the application and the limitations of the prototype hardware and software configuration. See Figure 1 for SIPS KSP configuration.

Three stages of analysis were established: Stage 1, initial data evaluation; Stage 2, comparison of initial and redo processing run data; and Stage 3, data trends. Files containing data quality and accounting information from processing runs on the GOULD, the SIPS mainframe, were downloaded to an IBM PC, acting as a dumb terminal, for expert system evaluation. The use of a database to store the data quality and accounting information as well as the decisions of each stage allowed the expert system to be divided into independent modules which run with the available memory of the prototype configuration.

The knowledge base of Stages 1 and 2 was developed with the



SIPS KNOWLEDGE SYSTEM PROTOTYPE CONFIGURATION

Figure 1

assistance of experienced QA/DA analysts and reference to existing QA/DA procedures. Stage 3, after initial investigation, was found to require a larger data sample set; this stage's development has been deferred in order to redirect resources toward an operational system, until a significant sample size is accumulated and statistically processed to derive data trends.

OPS5+ uses an "IF-THEN" format to represent knowledge required by the KSP for analysis. The rule interpreter implements a forward chaining technique on the input information to reach its conclusions.

The MicroSoft "C" compiler was used to create a user interface providing a menu of choices for entering initial information into the system database as well as for producing reports.

The PC/FOCUS software is a data management system package used for the validation of input data, the creation of databases, and statistical analysis.

2.2 THE SOPS ES PROTOTYPE

The SOPS ES Prototype was designed to perform the QA/DA analysis of SOPS data by isolating problem areas and recommending the appropriate course of action. The prototype was developed using the expert system shell CLIPS implemented on a Macintosh personal computer. The scope of the prototype was limited to the detail required only to realistically demonstrate the feasibility of an operational expert system. See Figure 2 for the SOPS ES prototype configuration.

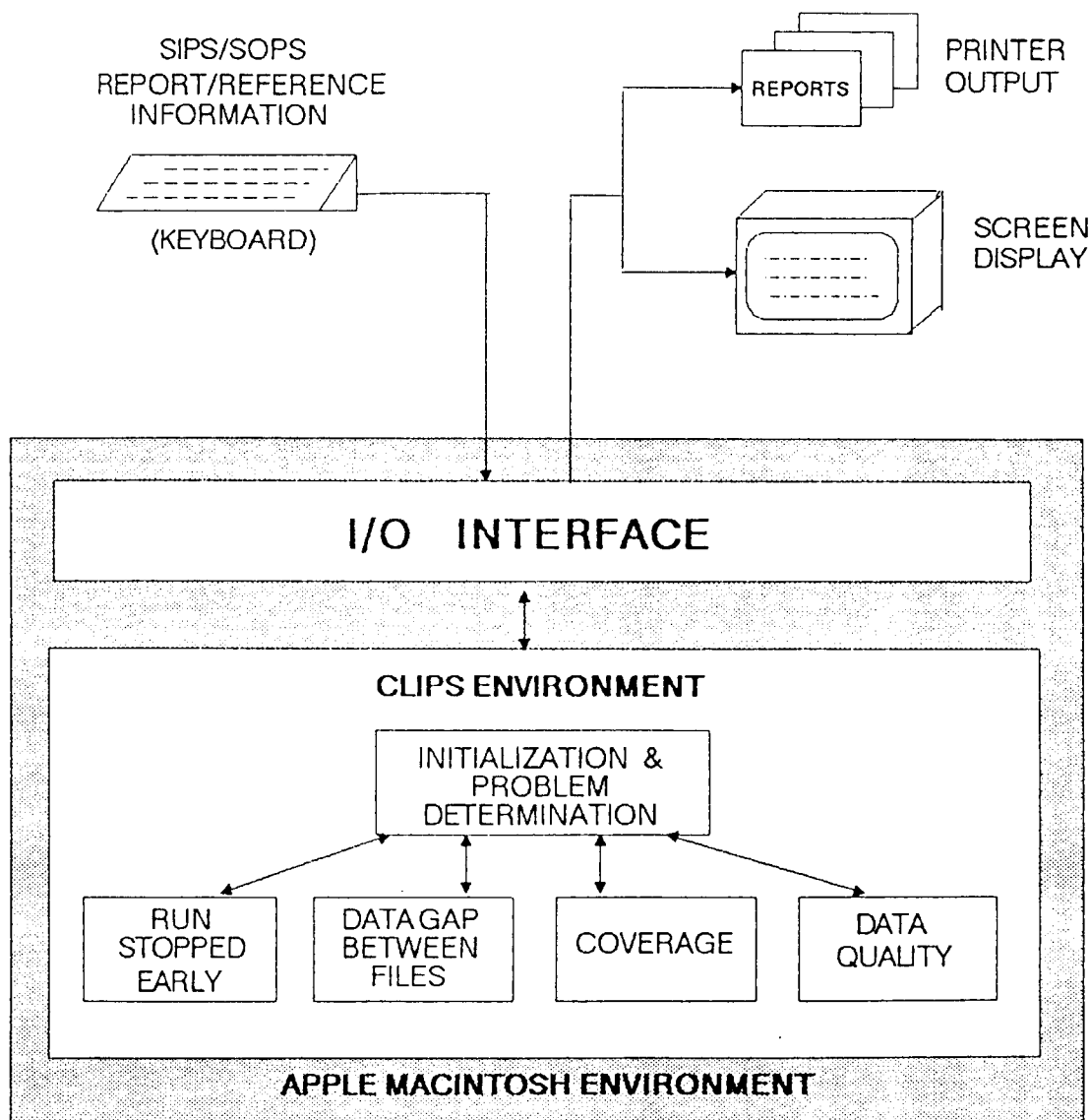
The knowledge base is composed of rules represented in the CLIPS form "IF <condition(s)> THEN <action(s)>." It can be logically divided into four major rule groups, each diagnosing a particular problem, driving the user interface, and retrieving information specific to that group: 1) Run Stopped Early, 2) Data Gap Between Files, 3) Data Coverage, and 4) Data Quality. (The initial data assessed by the ES prototype were simulated UNISYS 1100/82 SOPS application report files.)

The user interface of the SOPS ES prototype used many of the features that are standard for applications running on the Apple Macintosh. These features included the use of multiple windows, pull-down menus, and dialog boxes. See Figure 3 for a sample screen layout.

3. SLDPF RESPONSE TO PROTOTYPES

3.1 REALIZED IMPROVEMENTS

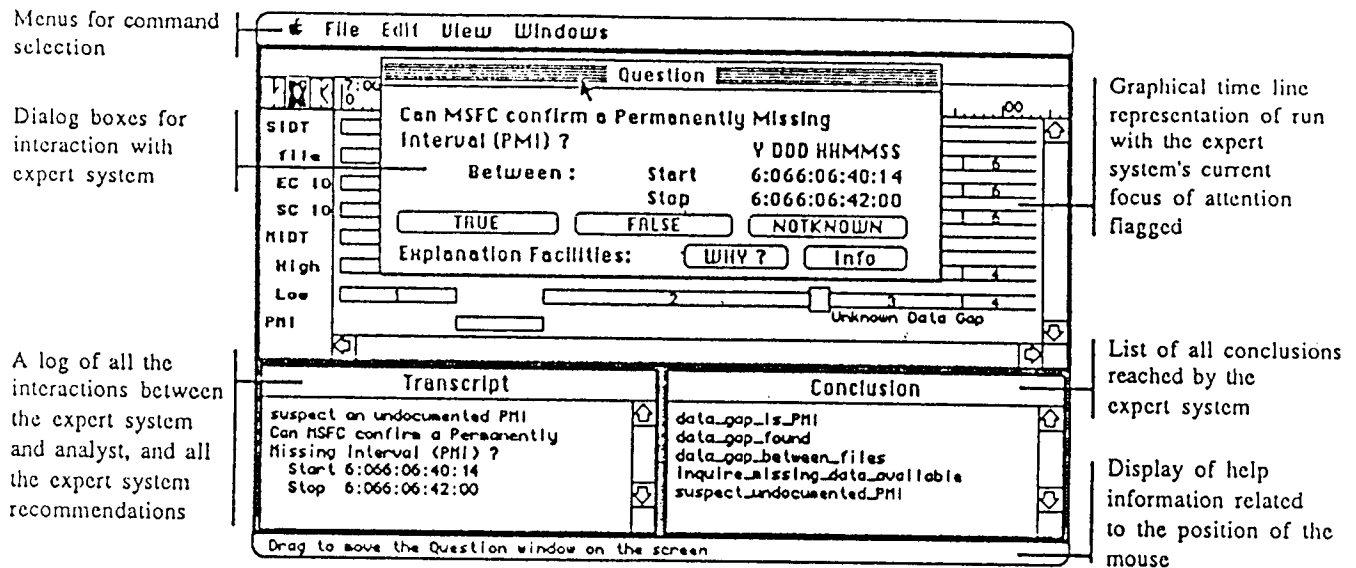
The reaction received after numerous demonstrations of the prototypes for operations and management personnel was quite positive. The prototypes exhibited speed and consistency in the data QA/DA evaluation. The physical demands upon operations personnel to analyze data processing runs were greatly decreased without diminishing the high productivity and quality standards



SOPS EXPERT SYSTEM PROTOTYPE CONFIGURATION

Figure 2

ORIGINAL PAGE IS
OF POOR QUALITY



SAMPLE SCREEN LAYOUT

Figure 3

which the SLDPF has consistently maintained.

Another benefit foreseen with the prototypes is the means of a training tool. Refining the dialog with the user to a more detailed level would allow a less experienced analyst to step through the logic of the expert systems and obtain explanations for each decision made. The expertise of the experienced staff members is captured within the expert systems and available when needed.

3.2 ADDITIONAL DESIRED ENHANCEMENTS

Although the prototypes already exhibited benefits over the previous manual QA/DA process, areas of improvement are being addressed for inclusion in the operational systems. The major concern was how to get the data which was to be evaluated onto the target workstations. The demonstrated methods of downloading data to diskettes or manual input defeated the purpose of an expert system by burdening operations personnel with another tedious task. This problem was solved with the proposal of a local area network (LAN) within the SLDPF. Data files containing the quality control and accounting information could be automatically transferred after the processing runs were completed.

Another issue was the necessity for the user to manually enter information for the expert systems which was only available on hand-written operations logs. It would be nearly impossible to eliminate all interaction with the user due to the realtime aspects of the SLDPF, but a majority of manual input could be avoided by storing the information in databases accessible to the expert systems. This process would not only increase the efficiency of the expert systems, but also automate some reporting functions.

Finally, the decision was made to develop the two operational expert systems on the same target workstation, using the same software packages, and developing similar user interfaces. The training of analysts who would most likely be working with both expert systems during mission support would be much more simplified. Also, some of the development and maintenance efforts could be shared between the two tasks.

4. OPERATIONAL EXPERT SYSTEMS

The goal of the ES prototypes was to determine the feasibility of expert systems in the mission environment. The necessary capabilities indicated by the prototypes dictated the design and configuration of the operational expert systems. It was determined that the systems would need to be more powerful and efficient than the prototype machines with the capability to interface with the current mainframes of SIPS and SOPS.

4.1 SYSTEM CONFIGURATION

The machine chosen to host the expert systems was the SUN 3/160 which meets all of the aforementioned requirements. The SUN 3/160 provides a general purpose computational engine with the power for Artificial Intelligence expert system applications along with window and graphic capabilities. It also supports the networking capabilities required for successful incorporation into the existing facility configuration.

The expert system tool CLIPS (C Language Integrated Production System), a forward chaining rule language, was selected to develop the expert systems. CLIPS was developed by the Artificial Intelligence Section (AIS) at NASA/Johnson Space Center. It was specifically designed to provide high portability, low cost, and easy integration with external systems. It provides reasonable performance on a wide variety of computers. A Users Help Desk is also available to provide documentation and software updates, and to respond to questions and suggestions.

The language chosen to develop the interface software was Objective-C, an object-oriented language. Objective-C is the C language grafted with a number of new syntactic features; it retains the efficiency and compatibility of C, but provides the reusability and productivity of an object-oriented programming language. Objective-C is a trademark of Stepstone, formerly PPI, who provides support for its licensed users.

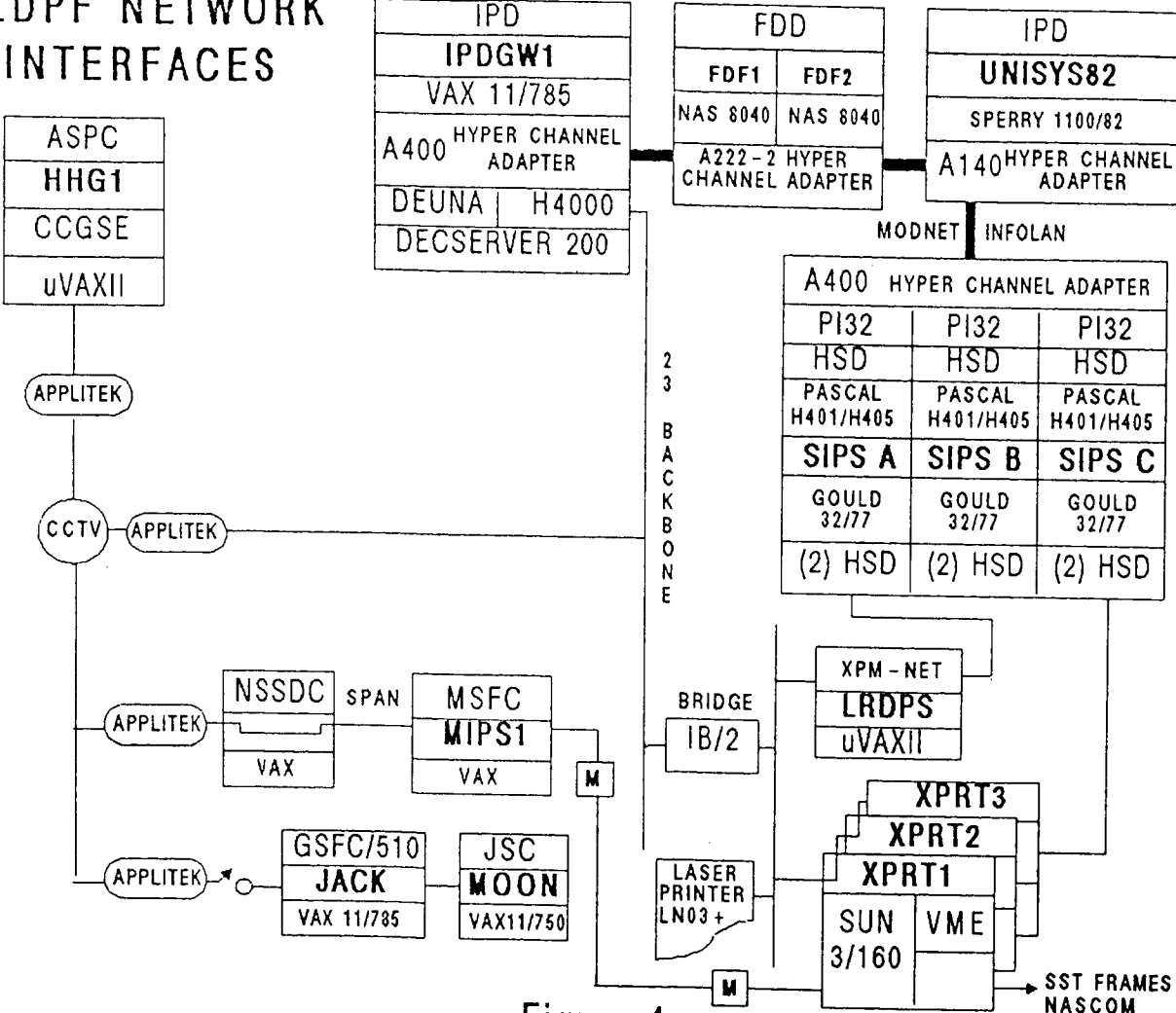
INGRES, a relational database management system, will be used to create and maintain the expert system databases. A comprehensive fourth generation language, visual forms editing, and host language interfaces are tools available to the application programmers. An additional advantage is that end-users can easily create simple queries, reports and graphs from existing databases.

A major strong point in the system configuration is the use of networks. See Figure 4 for the SLDPF Network Interfaces. The network connections within the SLDPF will allow communication and automatic transfer of data between the SLDPF mainframes and the SUN workstations. The SIPS GOULD 32/77 can transfer data over XPM-NET through a uVAXII onto the SLDPF LAN and to the SUN using DECNET. The SOPS UNISYS 1100/82 can transfer data using NETEX / BFX through the IPD Gateway VAX 11/785 onto the SLDPF LAN and to the SUN using DECNET. The main advantage of the networks is allowing easy access to the data quality and accounting information necessary for expert system evaluations without imposing numerous additional procedures upon the operations personnel.

4.2 THE SIPS ES DESIGN OVERVIEW

At the end of a SIPS processing event, the transfer of a file containing data quality and accounting information is automatically initiated from the GOULD 32/77 to the SUN Workstation. The SIPS ES validates the information, allowing the user to make any necessary adjustments, before the data is converted into its

SLDPF NETWORK INTERFACES



4/27/88
LB

object representations used in the Model Subsystem, and then stored in the ES database.

There are two ES evaluation options available for the user: Stage 1, initial data evaluation; and Stage 2, comparison of initial and redo processing run data. The ES creates a fact list for the selected data to be evaluated, loads the rule base and fact list into CLIPS, and executes the run. A summary report is automatically generated at the end of each run, as well as updating the database with ES generated parameters.

The components of the View Subsystem allow the user to interface with the ES. Information displayed on the screen can be modified to fit the needs of the user, and dialog with the ES is permitted through the View Subsystem.

See Figure 5 for the SIPS ES design.

4.3 THE SOPS ES DESIGN OVERVIEW

The Input/Output Edit and Decommulation (IOEDCM) and Experimenter Channel Data Edit (ECDEDT), the two main application programs within SOPS, create and store a run summary file containing quality and accounting information produced for each data processing run. A software routine within each application program will automatically initiate the transfer of this file from the UNISYS 1100/82 to the SUN Workstation.

The SOPS ES will read the transfer file and convert the data into the object representation used in the Model Subsystem. The Model Subsystem holds the run summary data and when instructed will copy the necessary data into the fact list for the inference engine to use. The inference engine maintains and modifies the fact list according to the rules read from the rule base. At the end of the ES session, a summary report is automatically generated and the database updated.

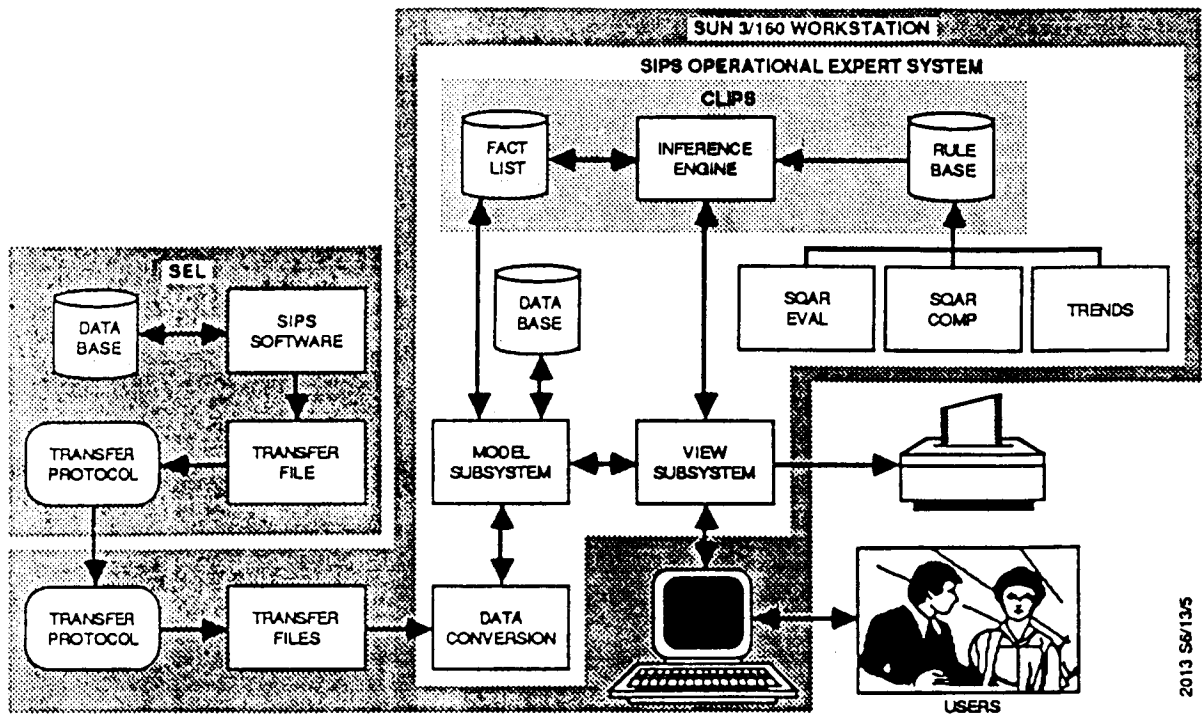
The View Subsystem functions as the interface between the analyst and the rest of the system. Through the View Subsystem the analyst can modify the information presented on the screen or direct the diagnoses of the inference engine.

See Figure 6 for the SOPS ES design.

5. ES MOTIVATED IDEAS WITHIN THE SLDPF

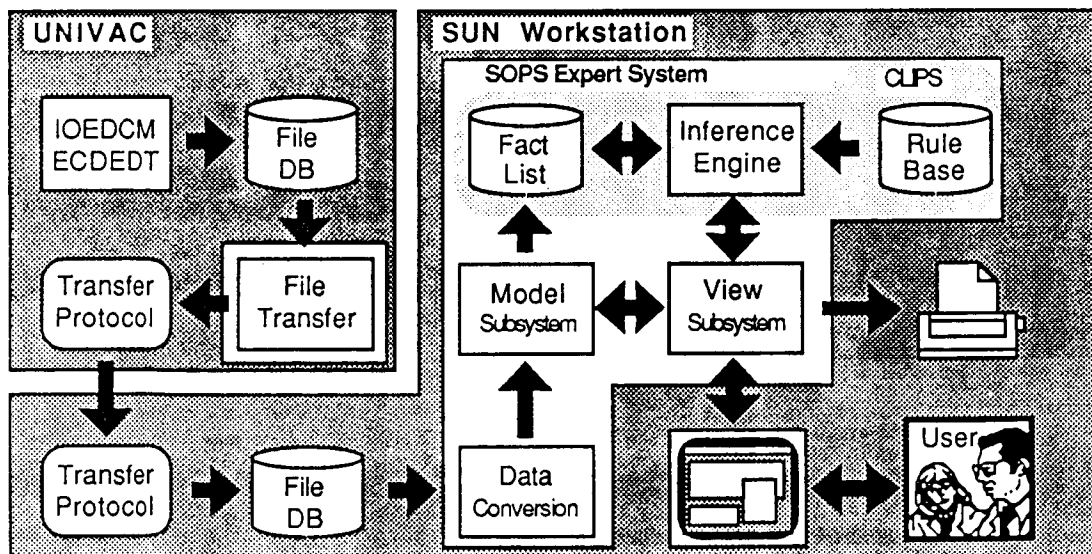
Along with the incorporation of the operational expert systems within the SLDPF configuration, other concepts are also envisioned to be implemented. First, a designated area within the SLDPF will be partitioned for the QA and DA analysts. This area will house the SUN Workstations and the other necessary equipment and space for the analysts to perform their tasks. Because of the networks, the analysts will have access to the SIPS and SOPS mainframe computers in addition to direct access to the expert systems and their database information. All work can be accomplished from one central area.

Secondly, because the expert systems will keep track of the SIPS and SOPS processing, it is only logical that many of the



SIPS EXPERT SYSTEM DESIGN

Figure 5



SOPS EXPERT SYSTEM DESIGN

Figure 6

ORIGINAL PAGE IS
OF POOR QUALITY

mission status reports can be obtained from their databases. This gave rise to the idea of a Centralized Information System (CIS) which would automate the reporting function that was in most part a manual effort of summarizing hand-written reports. Necessary information could be transferred from the mainframes to the SUN Workstations and stored in the INGRES database; also any information from hand-written logs would be entered. All this information along with the ES databases would be sufficient to automate the mission status reporting function.

6. SUMMARY

The prototypes have proven that expert systems offer many benefits. Throughout their development, ways have been identified to further automate current procedures, to increase accessibility to data, to improve processing speed, and to decrease the monotony of repetitious tasks. Subsequently, these concepts have also been carried through to other areas of the SLDPF. Not only will the work environment be improved, but also the outlook of the SLDPF personnel.

It is planned that this configuration will be operational by January 1989 in time to support the combined ASTRO-1/BXRT mission, the first of several scheduled SLDPF missions in the post-Challenger period.

REFERENCES

1. Ames, T., Spacelab Output Processing System Expert System Prototype Phase II. NASA Goddard Space Flight Center (1987).
2. Ames, T., Spacelab Output Processing System Quality Assurance and Data Accounting Expert System Design Document. NASA Goddard Space Flight Center (1988).
3. Watson, J., Dallam, W., and Ripley, W. D., Spacelab Input Processing System Knowledge System Prototype Final Report, L-05229. Lockheed Engineering and Management Services Company, Inc. (1987).
4. Watson, J., Ripley, W. D., and Dallam, W., Spacelab Input Processing System Quality Assurance and Data Accounting Expert System Design Document, CSC/SD-88/6017. Computer Sciences Corporation (1988).

AUTOMATED CATALOGING AND CHARACTERIZATION
OF SPACE-DERIVED DATA

William J. Campbell
National Space Science Data Center
NASA/Goddard Space Flight Center
Applied Artificial Intelligence Laboratory
Greenbelt, Maryland 20771

Larry Roelofs
Computer Technology Associates
McLean, Virginia, 22102

Michael Goldberg
MITRE Corporation
McLean, Virginia 22102

ABSTRACT

One of the most significant technical issues that NASA must address and resolve is the problem of managing the enormous amounts of scientific and engineering data that will be generated by the next generation of remote sensing systems such as the Hubble Space Telescope (HST) and the Earth Observation System (EOS). The amount of data these sensors are expected to produce will be orders of magnitude greater than NASA has ever experienced. Consequently new solutions must be developed for managing, accessing and automatically inputting the data into a database in some expressive fashion that will provide a meaningful understanding and effective utilization of this data in a multidisciplinary environment.

Presently, scientific data provided by satellites and other sources (i.e., in situ measurements) are processed, cataloged, and archived according to narrow mission or project-specific requirements with little regard to the semantics of the overall research. Scientists therefore lack knowledge of or access to potentially valuable data outside their own field. What is needed is an innovative approach that will allow collected data to be automatically cataloged, characterized and managed in a domain-specific context and made available interactively and in near-real-time to the user community. This paper discusses a concept and design approach that employs expert system-based knowledge controllers combined with advanced spatial database systems and graphical data structures.

PRECEDING PAGE BLANK NOT FILMED

AUTOMATIC CATALOGING AND CHARACTERIZATION OF SPACE DERIVED DATA

William J. Campbell
NASA/Goddard Space Flight Center
National Space Science Data Center
Applied Artificial Intelligence Laboratory
Greenbelt, Maryland 20771

Larry H. Roelofs
Computer Technology Associates Inc.
McLean, Virginia 22102

Michael Goldberg
MITRE Corporation
McLean, Virginia 22102

1. INTRODUCTION

One of the most significant technical issues that NASA must address and resolve is the problem of managing the enormous amounts of scientific and engineering data that will be generated by the next generation of remote sensing systems, such as the Hubble Space Telescope (HST) and the Earth Observation System (EOS). The amount of data these sensors are expected to produce will be orders of magnitude greater than NASA has ever experienced. Consequently new solutions must be developed for managing, accessing and automatically inputting the data into a database in some expressive fashion that will provide a meaningful understanding and effective utilization of this data in a multidisciplinary environment.

Presently, scientific data provided by satellites and other sources (i.e., in situ measurements) are processed, cataloged, and archived according to narrow mission or project-specific requirements with little regard to the semantics of the overall research. Scientists therefore lack knowledge of or access to potentially valuable data outside their own field. What is needed is an innovative approach that will allow collected data to be automatically cataloged, characterized and managed in a domain-specific context and made available interactively and in near-real-time to the user community.

2. BACKGROUND

2.1 Present Approach to Data Cataloging and Characterization

Presently, remotely sensed data are managed based on the data acquisition time/location history (i.e., where the sensor was looking when the data was collected). Such a data management approach permits the location of data sets only after first determining where one wants to look for a specific, scientifically interesting data object. The procedure used to manage data based on this approach is summarized as follows:

- Record the generated sensor data on magnetic tape and archive the tape.
- Create a database management system that manages information about the sensor data (i.e., create a data catalog). This catalog usually supports queries based on mission/sensor name, time of data capture, and location of data.

This approach is basically a mixture of manual and automatic processes that involves:

- Data correction and formatting
- Data verification
- Compilation of meta data (information about data)
- Inputting meta data into inventory of database
- Update inventory control of tapes and store tapes appropriately
- Reproduce data tapes for users as required.

The problem with the above approach is that it is both human time and computer process intensive in most instances. Consequently if the data sets are large or there are many data sets to be ingested, the system becomes overloaded and fails. Generally speaking, present procedures are sensitive to several important system considerations including:

- Database system software
- Database design
- Data and its supporting data structures
- Supporting data processing capabilities
- Data input processing time
- Human resources

Because the procedures tend to be system and project unique, the following two representative examples are provided to show how data catalog and characterization is presently performed by a specific project.

2.1.1 Crustal Dynamics Data Information System

In the case of the Crustal Dynamics Data Information System (CDDIS) ^[1], a user must submit a data tape (whose format has been standardized to a previously agreed upon common system format) to the CDDIS user support office which then performs the following:

1. Determines if the tape may need some correction or reformatting before further processing. If the tape is NASA-generated, then no corrections are needed,
2. Verifies the tape contents by electronically scanning the tape,
3. Summarizes the tape contents (i.e., summarizes data records by satellite pass and computing the total number of observations per pass),
4. Loads the summary into the appropriate database table (i.e., by year and by specific satellite),
5. Enters the tape into tape and file inventory,
6. Makes copies of the tape for the user,
7. Archives the tape in the CDDIS tape library.

This process generally takes from 10 to 20 hours over a period of three to five days.

2.1.2 Pilot Land Data System

In the case of the Pilot Land Data System (PLDS)^[2], the user must fill out an electronic or analog form which describes various attributes about the data set of interest and deliver this information and data to the User Support Office (USO) which then:

1. Unpacks the data set according to the specific format of each data type (i.e., MSS, AVHRR, etc.),
2. Depending on the preprocessing requirements submitted by the user, verifies the data for content, accuracy, and completeness or performs a preprocess such as first pass radiometric or geometric correction,
3. Performs steps 4 through 7 as described in the CDDIS above, except that for step 5, PLDS requires a more detailed description in the inventory.

In addition, a PLDS user may request (either at time of data entry or at a later date), additional information from a given data set such as generating a histogram, estimating the amount of cloud cover, or performing a prescribed subsetting operation.

As one can readily see, the above scenarios represent a tedious and cumbersome process that can take from three to five working days and when finally completed still provide the user with only restricted online information about any specific data set in question. The approach is self-limiting because only pre-conceived data can be selected based on a platform position and data capture time which the potential database management system user must know about or somehow determine. With only a few exceptions, data sets cannot be automatically selected based on a description of an interesting scientific attribute

(i.e., "vegetative index," "globular cluster") or a non-quantitative description of position (i.e., "Massachusetts," "Great Red Spot"). Therefore, these data sets are inaccessible by multidisciplinary or non-mission specific investigators.

2.2 Future Direction of Data Cataloging & Characterization

Projected Earth science research in the 1990's will require access to multiple data sets that have been returned from the various instruments on board the EOS platform as well as some that have previously been archived. Based on this requirement, the EOS Data Information System (EOS/DIS) must be developed to not only effectively store, manage, and support access to data, but somehow catalog and characterize it rapidly and efficiently. Based on estimated sensor data rates, data will be produced at a volume and complexity that exceeds the capabilities of all present and most emerging information science technologies and systems.

For example, it is proposed that data rates of at least 50 Megabits per second will be generated and somehow ingested into the EOS data archive, which then must be made available for distribution to users in hours or minutes (depending on a particular data set) ^[3]. Since such data rates exceed all but the most advanced information management and processing systems, new technologies must be developed that surpass the present limitations in data storage and management. However, given that such technologies are developed, information systems will still be unable to perform their mission properly because there presently is no way of cataloging & characterizing the sensor data into the system automatically. Consequently, we envision that all new information systems implemented to support next generation space based sensors must include not only technologies that store and manage the data, but also technologies that support intelligent, automatic cataloging & characterization of data into the system.

The first difficulty is a data storage and database management problem, which is being addressed and resolved at many levels both by government and private research and development activities. However, the second difficulty, automatic data characterization and cataloging, is presently being given little or no attention because it requires that a robust information management infrastructure already be in place, and data detection and identification tools be available to identify data objects in a data set as well as classifying and characterizing them in the context of some representative data world. Some preliminary research can be found in ^[4] and ^[5].

2.3 Related Impacting Research

Over the past three years NASA 's Intelligent Data Management Project (IDM) has been doing research to develop advanced data systems that can represent and manage both spatial and meta data (information about data) in a manner that will support the needs of scientists who have little or no understanding of database systems, (including their designs, data content and query languages), and provide the necessary architecture and software tools for supporting automatic data ingest into the database. The project has developed concepts, designs and demonstration prototypes using AI and information methodologies

and tools that address many of the issues involved in developing advanced database representation and interfacing such that a user with little or no experience with a particular database is able to communicate with the system effectively by using plain English or graphical input^[6].

As part of the IDM research activity, information system design concepts have been formulated and tested that allow a user to interact with a remote operational scientific database using natural language, graphics and expert systems that remove the burden of the user in understanding the database content, architecture and query language. This development extends database interfacing beyond the normal DBMS models such as the hierarchical, network and relational paradigms. Basically, these models provide three necessary features required for performing information management:

1. Standard facilities for describing the logical structure of a database (using trees, collections of nodes and links, and relations (tables), respectively);
2. General access processes using a data manipulation language;
3. Update operators (language primitives) for manipulating these data structures.

Although all three DBMS models are presently in use today, IDM research activities focused on using the relational model for storage of meta data because of its flexibility, adaptability, and robust language facilities. Recognizing that the unnatural representation forced on the data to facilitate efficient database search processing destroys the semantic content which the user hold paramount, two important problems become immediately apparent. First, the database is very difficult to use (except to the most experienced database users) because of the database architecture and data naming conventions that are imposed, and second, inputting data into the database is difficult to perform because there is no logical connection between the data being stored, the data structure used to support it and the data source.

As a consequence, our research activities have concentrated on adding back into the information system the meaning of the data in context and devising a logical database representation schema that is both application and context dependent. In addition, a language facility has been added that allows the user to interact with the database using English-like expressions. With such a capability users will be able to find data in a logically straightforward manner.

3. PROPOSED SOLUTION

What is needed is an automatic data cataloging & characterization approach that deals with the problem in a single end-to-end integrated environment starting with the sensor and ending with the production of data suitable for inclusion in a scientific publication. The approach is to create meta and spatial database "search keys" that characterize and summarize incoming data with respect to the needs of the scientific community.

3.1 Automatic Data Cataloging and Characterization (ADCC)

As conceptualized, the automatic cataloging and characterization of new datasets into a operational database is depicted in Figure 1, Top Level Architecture for Automatic Cataloging & Characterization, discussed in the following paragraphs:

1. A new dataset is sent to the information system from a sensor via some intermediate processing center where data decompression and calibration have been performed and proper data set identification has been added, such as date, time and sensor ID. This is called the initial meta data processing level.
2. This data is then sent to the appropriate archive and its supporting meta data is processed and proper data set identification is determined. A reference data frame that is specific to a particular domain (science or sensor specific) is then created within the context of the domain world model of the information management system. An important point concerning this process is that much of the information required to catalog and characterize the data set is already in the knowledge base as a consequence of a priori knowledge acquisition from both the ephemeris information specific to a given sensor as well as the science information a particular sensor is designed to capture.

For example, one of the scientific objectives of the Moderate-Resolution Imaging Spectrometer (MODIS)^[7] is to determine continental changes in snow cover with associated changes in albedo. The process would determine that the data stream is from the MODIS instrument and route that information directly to the frame specifically designated and customized (sensor specific information as well as science information), for MODIS. Other frames will be engineered specific to other EOS instruments (SAR, HIRIS, etc.). After the sensor header information is cataloged, a high level data characterization agent will be automatically activated to determine if any measurable changes occurred in albedo from any previous frame. If a change is detected, this information will immediately supersede any previous albedo information. This scenario will allow a user interested in albedo changes acquired from MODIS to efficiently and rapidly query the system and obtain that information that is specific to his research without having to go through a lengthy browse and database interrogation process.

3. This process will continue for all of the instruments and sensors specific to EOS (or for that matter any other NASA mission), by having each data stream analyzed by identification, classification and characterization agents which will be controlled by a knowledge-based planner and controller that directs the identification and abstraction of high-level data objects using the appropriate domain-specific program. The actual characterization of the high-level data could be activated by a neural network approach which is a dynamic system using a topology of a directed graph which would process information by means of a state response to continuous or initial input.

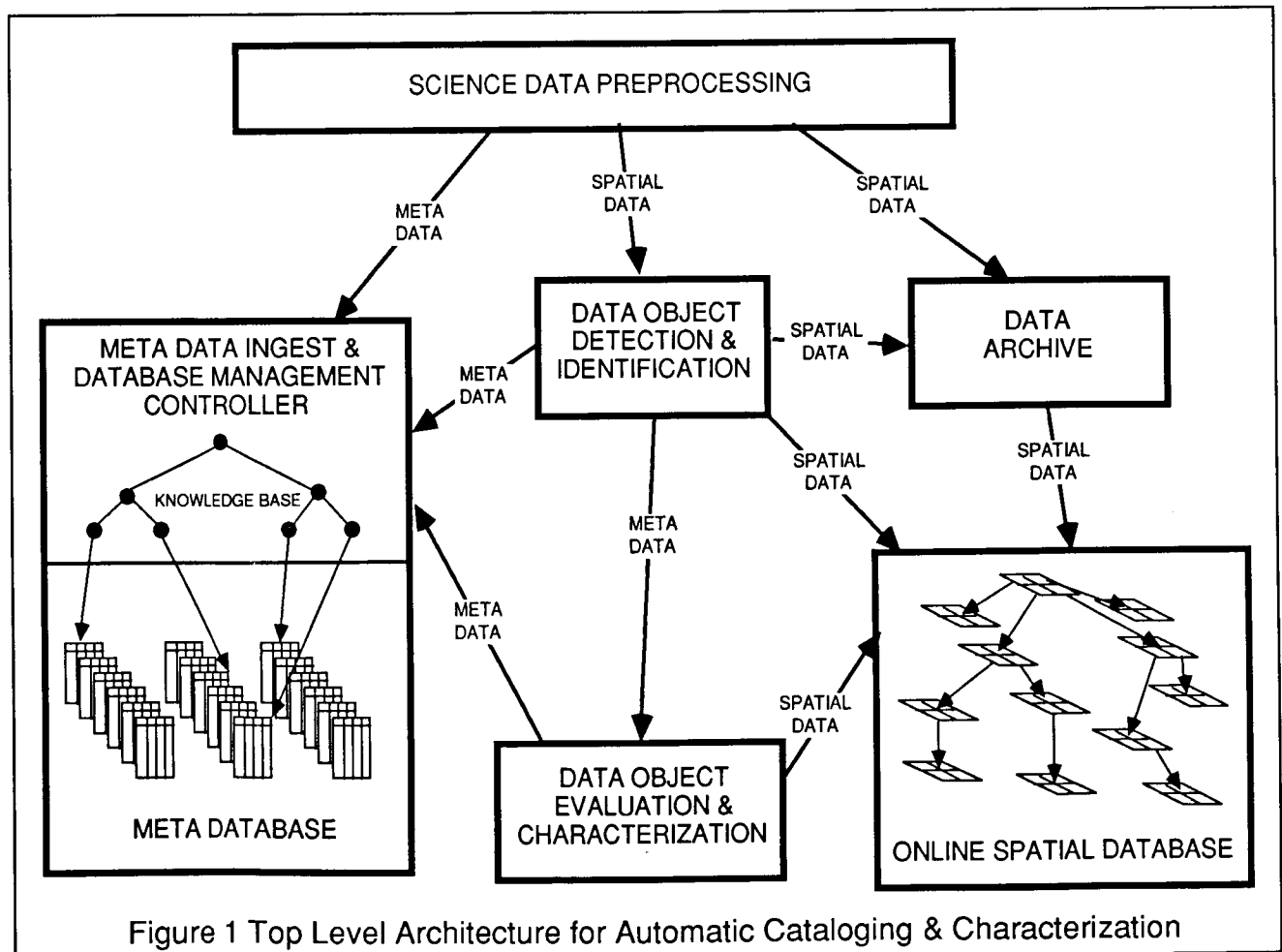
Using the above approach, raw science and engineering data can be efficiently processed and stored using meaningful representations that are more suitable to a user's reasoning. The definition, development, and evolution of the meta frames, agents and overall data system model are the first steps in the evolution of an application-driven knowledge base.

3.2 Design Considerations

The design of a data cataloging and characterization system is predicated on having preexisting knowledge of the domain, the sensor devices and the interpretation of their measurements. It is fruitless to identify, store, manage and ingest data if there are no guidelines that differentiate between good and bad observations, or if the integrity of the database cannot be guaranteed.

The suggested design employs expert system-based knowledge controllers combined with advanced spatial database systems and graphical data structures. This design would require a research and development effort in the following areas:

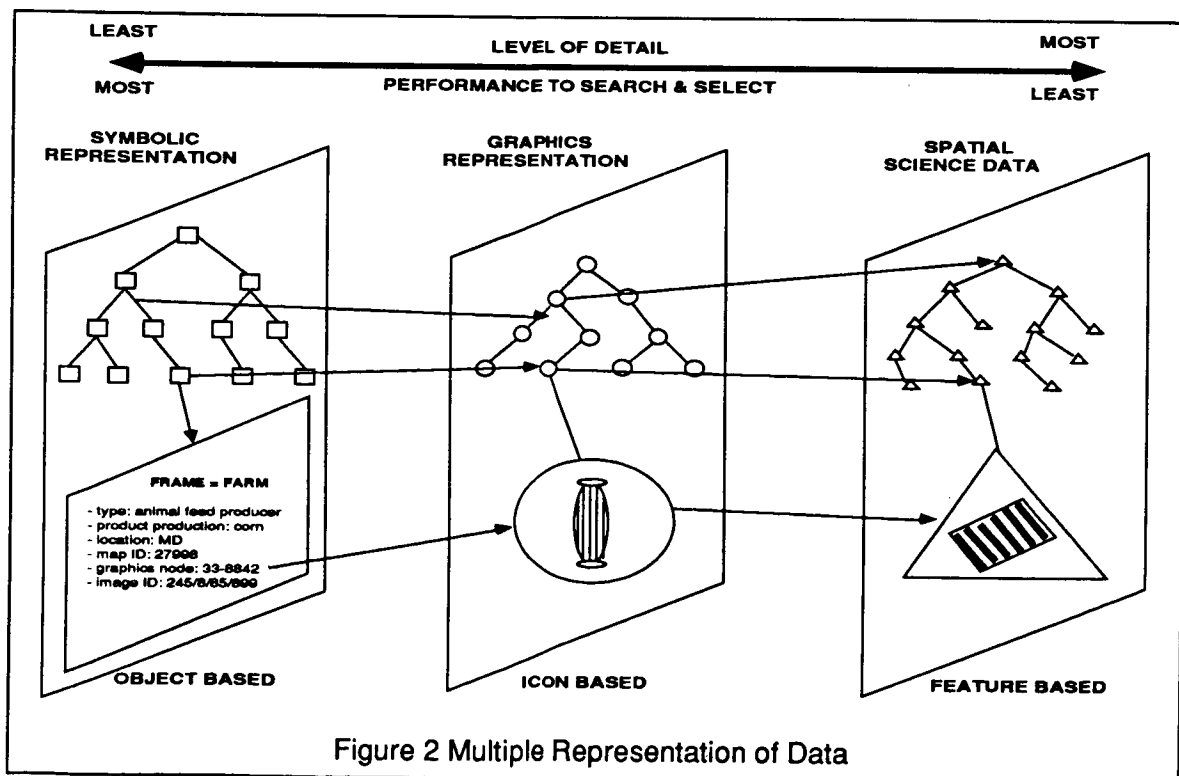
1. Logical data structures using expert and spatial database systems that maintain context and are interconnected.



2. Intelligent processes for determining data context in a data-world domain.
3. Intelligent processes that allow the identification of dataset attributes in the incoming datasets.
4. Graphical representations of data objects that can summarize the representation of data and information in spatial context but which is still tagged to the meta data. This concept is shown in Figure 2, Multiple Representation of Data.
5. A planning system that is used to supervise the initiation and operation of cataloging and characterization agents based on a prescribed goal, such as a specific research or mission objective.
6. On-the-fly network display indicating the current status of system maintenance as well as the flagging of any anomaly in a dataset.

3.3 Design Issues

1. Position of ADCC within existing NASA database management system environments. ADCC could potentially be incorporated within existing NASA database management environments as an addition to Level One Processing, as a function of the proposed Customer Data Service Facilities, as a function of the National Space Science Data Center, or as some combination of the above. These options must be evaluated and selected because they affect the development strategy and ultimately the success of this essential new concept.



2. **Development Methodology.** Because ADCC is dependent on relatively untested technology, it must be developed and incorporated into existing systems through rapid prototyping and testbedding methodologies rather than through traditional system integration approaches. A strategy for migration from testbedding to operations should be developed within this context.
3. **Synergy with Other Aspects of Science Data Management.** It both drives and is driven by the design of science data capture, scientific data analysis, data storage, data transmission, and the user interface. Although ADCC is presented as a separate topic, it should be designed and implemented as an interdependent subsystem of an overall intelligent science data management system.^[3]

3.4 Implementation Costs/Benefits

Inherent in the resolution of issues 1-3 above is a consideration of the tradeoffs between the costs (development, operations, maintenance, etc.) and the benefits (operational, scientific, etc.) that would be involved depending upon which design and implementation options are taken. Although the development of ADCC is being proposed, some consideration of cost/benefits should be made to guide this development.

4. RECOMMENDATIONS

Recommendations for the development and implementation of designing, developing and implementing an automatic data cataloging and characterizing capability fall into three categories: technical, institutional and test bedding. In the following subsections each of these areas are addressed.

4.1 Technical

1. ADCC is essential to providing scientists and engineers with fast access to the data they need and with a multidisciplinary database to choose their data from. ADCC, along with the other components of intelligent science data management, should be pursued as a viable technical approach to NASA's data management problems.
2. The design of an ADCC must be consistent with: (1) scientists' data analysis and user interface requirements, (2) experiment data collection (including onboard data reduction and expert-system directed data capture), (3) satellite and ground data formats, (4) Space Station era data storage and interchange structures, and (5) data analysis requirements. All of these components must be integrated as part of an overall science data management system.
3. ADCC should take advantage of relevant artificial intelligence tools, information management and data storage technologies, advanced spatial database systems, graphics, and data structures as appropriate.

4.2 Institutional

1. Improved techniques and standards for managing Space Station era scientific data, and specifically those techniques and standards that speed users' access to data and increase the multidisciplinary use of data, should be of the highest priority to future NASA data management systems.
2. A vigorous program for testbedding ADCC and other related science data management capabilities should be pursued.
3. ADCC should be implemented as part of Level One and beyond data processing.
4. ADCC should be implemented for domain specific data base management systems, and large multipurpose data archive systems, as appropriate.
5. The knowledge base and control processes must be exportable to distributed archives and user data centers along with the data.

5.0 Testbedding

The following testbeds related to ADCC should be pursued in a scientific applications environment:

1. High-performance graphics capabilities for display and analysis of spatial data and conceptual data base views,
2. Hierarchical data structures for representation of spatially referenced data bases and meta-data knowledge bases in a manner that minimizes data search time yet maintains database detail and integrity,
3. "Intelligent controllers" to direct the automatic cataloging and characterization of information from spatial data sources (i.e., images) as well as meta data sources (i.e., ancillary data files) and to direct the subsequent entry of this information into meta database catalogs and inventories,
4. The development of data object detection, identification and characterization agents that can be directed by the intelligent data management controllers to evaluate new meta data in context of the specific domain,
5. Benchmarking of improvements in data access speed and range of multidisciplinary queries,
6. User interface development using natural language and other interface paradigms in conjunction with graphics capabilities to review data as they are ingested.

6. SUMMARY

The successful development and implementation of the ADCC concept would have a direct and significant impact on scientific research. It would allow NASA for the first time to develop and build information systems that not only allow users to efficiently find data that is required for their research but will intelligently recognize and extract higher-level objects embedded in the raw data in near-real-time. Based on present estimates of future sensor data rates and the expected cost for analyzing such data, an information management scheme that includes automatic data cataloging and characterization at the lower level appears to be a highly viable alternative.

In addition, a research and development effort should be initiated that is coupled to work being conducted by the Intelligent Data Management Project, Visualization of Scientific Data and Distributed Access View Integrated Database (DAVID) ongoing in the National Space Science Data Center (NSSDC). The fusion of these technologies will allow the resulting information derived from these processes to be easily displayed at a remote users site in a fashion most useful for scientific inquiry.

7. ACKNOWLEDGMENT

The authors would like to thank Robert Crompt and Scott L. Wattawa of Science Application Research Incorporated for their critical inputs and review of this paper as well as Nicholas Short Jr. of the National Space Science Data Center for his support and technical guidance.

8. REFERENCES

- [1] "Quick-Look Guide to the Crustal Dynamics Project's Data Information System", Carey E. Noll, J.M. Behnke, H.G. Linder, NASA Technical Memorandum 87818, June 1987.
- [2] "Advancements in Land Science Data Management Pilot Land Data System," William J. Campbell, P.H. Smith, R. Price, L. Roelofs, *The Science of the Total Environment*, 56 (1986) 31-44, Elsevier Publishers.
- [3] From Pattern to Process: The Strategy of the Earth Observing System, EOS Science Steering Committee Report, Volume II, National Aeronautics and Space Administration, 1987.
- [4] R.P. Gorman, T. Sejnowski, "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets," *Neural Nets*, (1988), 1, pp. 75-89.
- [5] R.S. Michalski, R. Stepp, "Learning from Observation: Conceptual Clustering" in Machine Learning eds. R. S. Michalski, J.G. Carbonell, T. M. Mitchell, 1983, Palo Alto: Tioga Publishing Company, pp. 331-363.
- [6] "The Intelligent User Interface for NASA's Advanced Information Management Systems," William J. Campbell, N. Short Jr., L. Roelofs, S. Wattawa, of "Third Conference on Artificial Intelligence for Space Applications, Nov. 1987, Pg. 359 Part II.
- [7] MODIS, Moderate-Resolution Imaging Spectrometer Instrument Panel Report Volume IIb Earth Observing System, National Aeronautics and Space Administration, 1986.

**A DESIGN FOR A GROUND-BASED
DATA MANAGEMENT SYSTEM**

Barbara A. Lambird
David Lavine
L.N.K. Corporation
6811 Kenilworth Ave.
Riverdale, MD 20737

ABSTRACT

An initial design for a ground-based data management system which includes intelligent data abstraction and cataloguing is described. The large quantity of data on some current and future NASA missions leads to significant problems in providing scientists with quick access to relevant data. Human screening of data for potential relevance to a particular study is time-consuming and costly. Intelligent databases can provide automatic screening when given relevant scientific parameters and constraints.

The data management system would provide, at a minimum, information on availability of the range of data (e.g., spectral range), the type available (e.g., LANDSAT, SPOT), specific time periods covered together with data quality information (data gaps, instrument problems, etc.), and related sources of data. The system would inform the user about the primary types of screening, analysis, and methods of presentation available to the user. The system would then aid the user with performing the desired tasks, in such a way that the user need only specify the scientific parameters and objectives, and not worry about specific details for running a particular program.

The design contains modules for (1) data abstraction (including spatial databases), (2) catalog plan abstraction, (3) a user-friendly interface, and (4) expert systems for data handling, data evaluation and application analysis. The emphasis is on developing general facilities for data representation, description, analysis, and presentation that will be easily used by scientists directly, thus bypassing the knowledge acquisition bottleneck. Expert system technology is used for many different aspects of the data management system, including the direct user interface, the interface to the data analysis routines, and the analysis of instrument status.

The system design uses the NASA-developed expert system development tool CLIPS and will be implemented on a Sun 3 workstation. The work is being supported by a NASA Small Business Innovation Research Phase I contract NAS5-30280.

INTRODUCTION

We will describe some key components of a ground-based data management system that will combine several computer technologies including third generation expert system tools, advanced data structures, spatial, graphical and other scientific databases, and pattern, image and other kinds of scientific analysis. Past and current space-based research is generating enormous amounts of many different kinds of data, that is presently stored primarily on magnetic tape. Examples include astronomical data, astrophysical data, atmospheric data, ionospheric data, land science data, magnetospheric data, ocean science data, planetary data, and solar-terrestrial data. In many cases, each data set is largely inaccessible to all but the few scientists who designed and built the original instrumentation, and even for them the data may be cumbersome to use.

The data management system we are designing and implementing will include data screening, browsing, and low and high level data analysis capabilities. The system will initially be applied to several problems in remote sensing using multispectral data. The system will include user models and efficient search techniques to enhance the efficiency of inferencing and spatial database access. Artificial intelligence-based techniques will provide for more automated intelligent browsing and integration of data. Much tedious data handling, such as formatting, preparation and presentation will be handled automatically by the system. Finally, the system is being designed to be easy for the scientist to use directly, without a detailed knowledge of the expert systems, thus reducing the typical knowledge acquisition bottleneck, and making data analysis techniques more widely available.

The data abstraction system is envisioned to be part of a larger distributed data management system, as shown in Figure 1. The distributed system would consist of a series of relatively inexpensive workstations networked with larger computer systems such as VAXes. The network need not be local, but could also be distributed over broad area networks.

The workstation would provide a uniform method for interfacing with the data management system, and would have extensive graphics and display capabilities. Expert systems for user models, data screening and browsing, data analysis and data presentation would reside on the workstations. The larger computer systems would contain much of the actual data analysis software and most of the data. Optical disk technology would be used to provide fast access to the large amount of archived scientific data. In the past, special purpose systems for artificial intelligence have been the predominant type of workstation used for the proposed type of work. The current trend is towards general purpose systems. The demonstration system is being implemented on a Sun workstation.

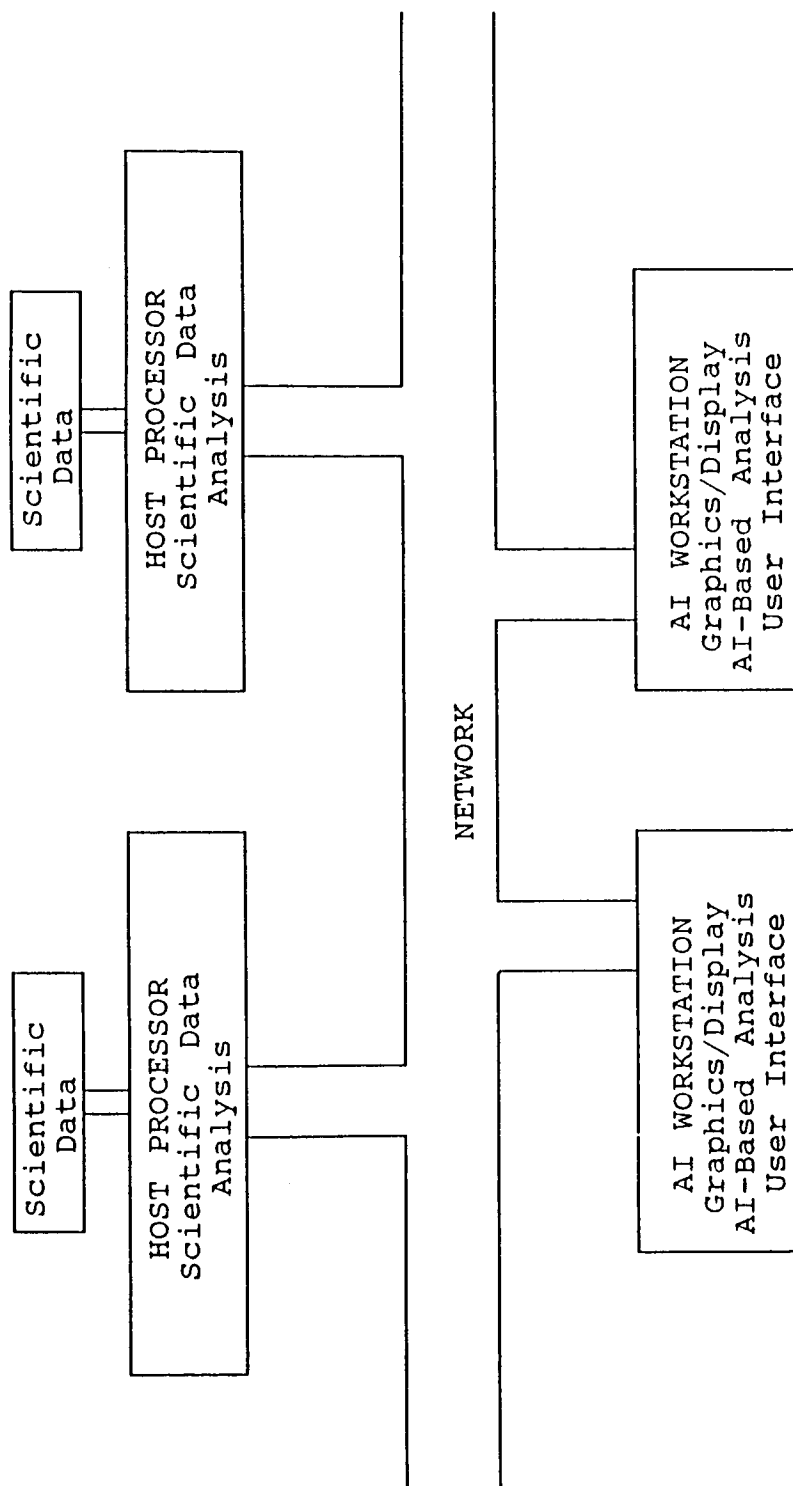


Figure 1. Configuration of the Data Management System.

The purpose of the system is to allow a scientist to sit at a workstation without any previous knowledge about the types of data, software and computers available, and interactively solve his problem in terms of scientific goals. For example, suppose the scientist wishes to study the effects of drought on vegetation in a particular area. First, the scientist chooses terrestrial remote sensing from a broad selection of scientific areas of inquiry. The system then provides more information about terrestrial remote sensing. From this information, the scientist chooses multispectral analysis. The system then provides more information about the types of multispectral analysis available. The scientist can either choose the spectral bands of interest from personal theory or be guided by the system from previously stored scientific goals. The process continues until the system has identified all the processing steps and parameters needed at an abstract level. The scientist is isolated from the actual details needed to perform the analysis tasks. At this point, the estimated cost of performing the analysis is provided to the scientist, who can decide whether to proceed.

The proposed data management system would be able to provide, at a minimum, information on availability of the range of data (e.g., land remote sensing, solar, magnetospheric, etc.), the type available (e.g., Landsat for remote sensing), specific time periods covered, together with data quality information (data gaps, instrument problems, etc.), and related sources of data. The system would be able to inform the user of the primary types of screening, analysis, and methods of result presentation already available for each data set, or aid the user with constructing new techniques using general artificial intelligence-based tools. The system then aids the user with performing the analysis, so that the user only need specify scientific goals, parameters, and display options, and not worry about specific details for running a particular program on a particular computer using a particular program under a particular operating system. If new data analysis methods are constructed, they would be assimilated into the system and made available to other users. In addition, the system would estimate the costs to the user (both in terms of time and money) of analyzing the data given the desired goals and parameters.

In summary, the user is not interested in where the data resides, or that the particular computer and software needed to perform the screening may be several thousand miles away. What is of interest to the scientist, is that the desired analysis can be accomplished within a certain amount of time at a certain cost.

Expert system technology will be used for many different aspects of the data management system, including the direct user interface, and the interface to the data screening in order to insulate the scientist from the software and hardware. Current

expert system technology provides mechanisms for efficiently building and maintaining complex systems which must "imitate" some aspects of human reasoning. In the case of data screening, for example, the user must choose a scientific goal, select parameters, select a data set, and construct the sequence of tasks to screen the data. These actions can be captured using "if - then" type of reasoning, which is the basic knowledge representation provided by most expert systems.

The other type of knowledge representation used by many expert system tools is object oriented programming. Object oriented programming is a technique in which information is usually stored in a hierarchical, modular form such that all information about a particular item or action is stored together. Unlike more traditional techniques, object oriented programming also provides a mechanism called methods for storing procedural information along with the factual information. In other words, "how to form or use the item" is stored along with factual information about the item. A typical implementation of object oriented programming is frames [Nilsson 1980]. A frame may be thought of as a group of "slots" each of which can store factual information, procedural information or point to other frames. Slots, in some implementations, can include explanations, defaults, and methods for producing the information when it is missing.

SYSTEM DESIGN

There are many different kinds of knowledge or data in the data management system: knowledge about the sensors and their data sets, knowledge about the computer hardware and software, knowledge about data analysis techniques, scientific knowledge, and knowledge about the users. An intelligent data management system must be able to represent and use all of them, which is discussed in more detail in this section on system design.

Instrumentation Hierarchy

There is a great deal of knowledge about the sensors. Some of this knowledge is shared with other instruments aboard the same platform (e.g., satellite, aircraft, balloon, etc.), such as flight or orbit parameters. Other information is specific to an instrument. Many instruments are actually a cluster of sensors which share similar or related scientific goals and hardware. Many of these individual sensors may run in several or many modes.

In order to efficiently represent the various aspects of the instrumentation, an instrumentation frame hierarchy will be used (see Figure 2). Information applicable to the whole platform is stored at the highest levels; information specific to sensors is included at the sensor level, and so on. The purpose of the hierarchy is that information is stored only once, i.e., information at higher levels is applicable to all lower

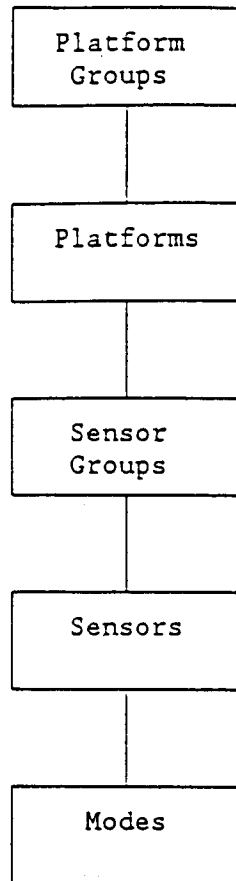


Figure 2(a). The Instrumentation Hierarchy that is part of the knowledge representation of the data management system.

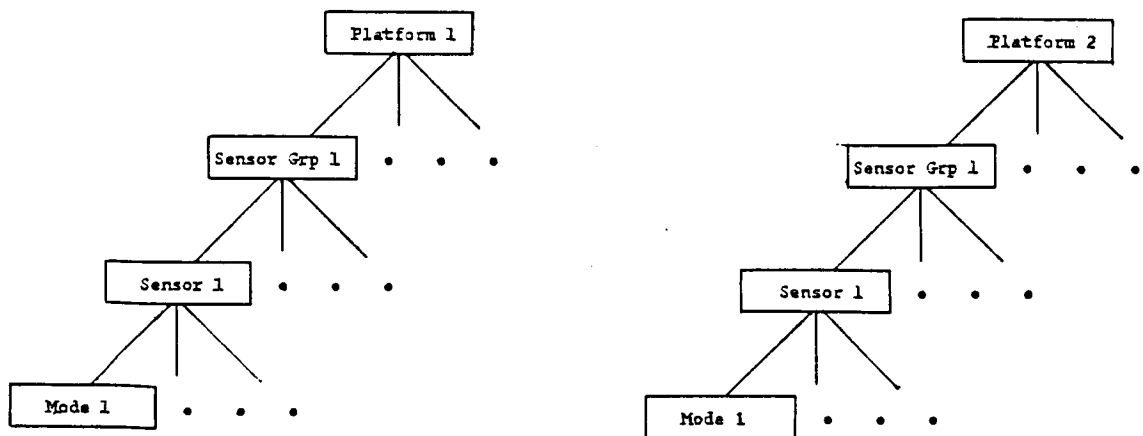


Figure 2(b). An example of an instantiation of the Instrumentation Hierarchy.

levels, and need not be repeated at the lower levels. Thus information related to the platform as a whole is stored once and not repeated for each individual sensor aboard the platform.

The top levels of the instrumentation hierarchy contain slots for data that is general to the entire platform. Examples include orbit or flight parameters, time periods covered by the platform, and status of the platform. These goal frames contain the scientific objectives for the platform as a whole, and are therefore relatively general. Procedural slots for determining information about the platform as a whole are also provided. The platform frames also contain a set of pointers to the sensor groups aboard the platforms.

The frames for each sensor group contain slots for information specific to the sensor group and shared by all the sensors within the group. Examples of this information include group status, time periods in operation for the group, and sensor group operation parameters. Information and procedures needed to interpret the data stream specific to this sensor group would also be stored in slots at this level. The sensor group frames also contain pointers to scientific objectives for the sensor group stored as frames in the scientific inquiry hierarchy. The sensor group frames also contain a set of pointers to the individual sensors included in each sensor group.

The contents of the frames for each sensor are analogous to the contents of the frames for sensor groups, except the information is specific to the individual sensor. Information at this level includes relevant engineering data such as frequency bands of multispectral scanners, and procedures for preprocessing raw data for purposes of rectification. Again, scientific goals specific to the individual sensors are provided through pointers. The sensor frames contain a set of pointers to the different modes of operation of each sensor. In the case of sensors with a large number of modes, an AND/OR graph [Nilsson 1980] of parameters used to generate the modes can be used for efficiency purposes.

Finally, if there is more than one platform working together, such as several satellites with inter-related projects, then the hierarchy would have an additional level on top, generalizing the lower level.

Slots which contain pointers to data sets associated with the sensors can occur at various levels. For example, status information which occurs at different levels is time-varying and is referenced through pointers. Information about the location of the actual data sets associated with the sensors is obtained through pointers to the data archive catalogs.

Scientific Inquiry Knowledge Base

The Scientific Inquiry Knowledge Base (SIKB) contains a

hierarchical description of common types of scientific information which a user may want to extract from the system. The SIKB is organized by scientific subject matter (e.g. land use classification, hydrological modelling). The SIKB is used to form plans to use existing software to answer scientific queries and to store the resulting plan. The answer to a query will often involve the use of several well known analysis techniques, with some possible variations, and in a relatively fixed sequence. At present there is massive duplication of software to perform these operations among scientists around the world. The SIKB allows the user to express his problem in terms with which he is familiar and let the system do the work of finding relevant software. Initially, the complexity of the scientific queries may be limited in scope by the limited number of tasks in much current data analysis. As the system is used more and a wider range of data analysis tools become available, the breadth and depth of the analyses by the SIKB will grow.

The SIKB will be organized as a set of knowledge hierarchies in different scientific fields. Each hierarchy will be an AND/OR tree with an AND/OR subtree representing the specific problem to be solved. The user will generally traverse a path in the SIKB until a problem to be solved is reached. At this point, the section of the SIKB below this problem provides information about subgoals involved in solving the problem and steps to be taken to achieve these goals. Depending on the availability of previous work, there may be many choices for the user at this point as to how to meet the subgoals. The SIKB uses the Analysis Toolbox, discussed in the next section, to decide what types of operations will be used to meet goals. For example, the SIKB may be given a query about vegetative cover in a portion of a Landsat image. One step in answering the query may be to perform pixel-based classification of vegetative cover. When the data analysis plan has been developed to the point where it is known that this classification is required, the SIKB will access the Analysis Toolbox to determine potential relevant classification algorithms. A highly simplified example of a portion of a sample SIKB is given in Figure 3.

An important part of the knowledge stored in the SIKB is the relationship between different kinds of data in terms of achieving various scientific objectives. There are two kinds of relationships. The first type of relationship concerns interchangeability of data. This type of relationship would specify that for a particular application, two data sets may be interchangeable. For example, if the user is interested in relatively large vegetative features, then data of several different resolutions may be adequate for the analysis. The second type of relationship specifies that two data sets can be used together more effectively than either could alone for a particular application. For example Landsat data can be effectively combined with elevation data to perform more sophisticated hydrological modelling than could be done with either alone.

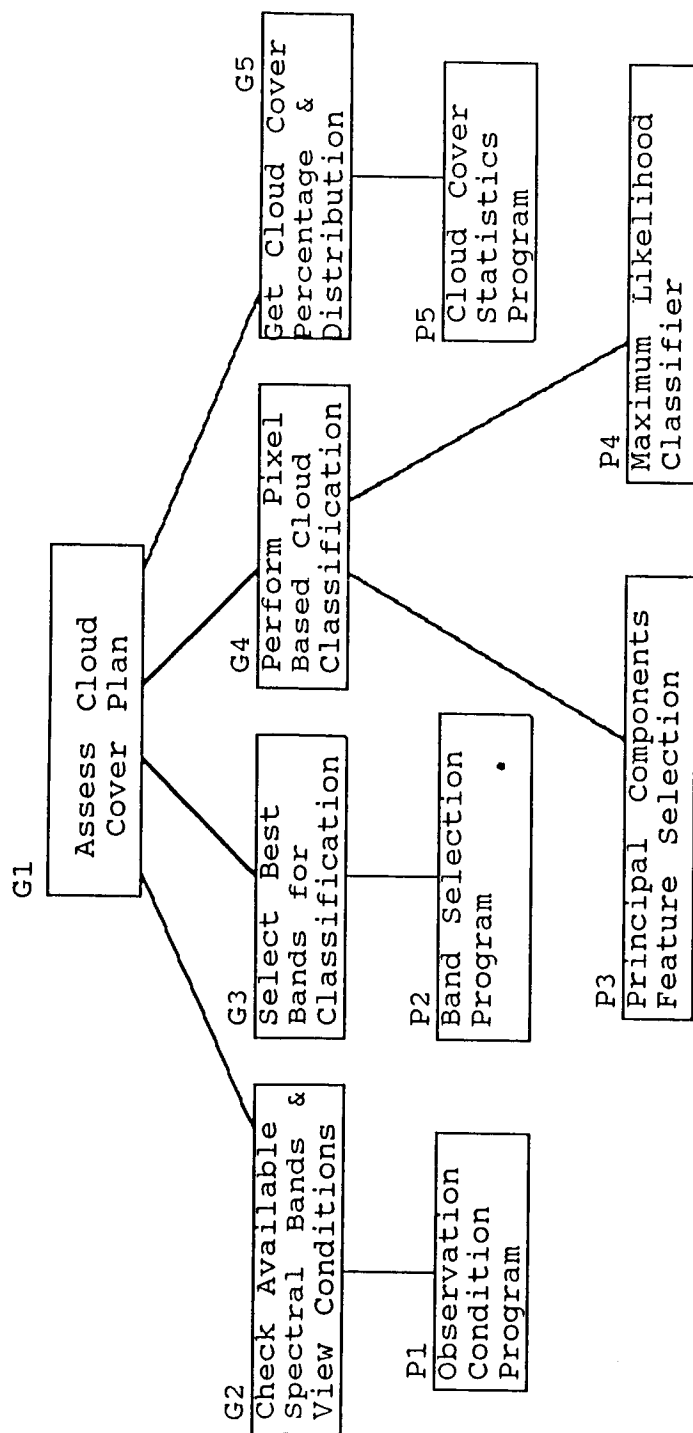


Figure 3. Simplified Cloud Cover Assessment Plan.
 G1 through G5 represent goals; P1 through P5 represent procedures.

Analysis Toolbox

The Analysis Toolbox (AT) contains network descriptions of a wide variety of analysis tools for tasks including numerical methods, statistical analysis, linear systems analysis, image analysis, signal processing, and data presentation. These tools are used in many diverse applications and it is important that the scientist be able to browse through and select from these tools, even if the tools have not yet been associated with the application of interest to the scientist. The tools are functional tools for which software packages exist. A tool in the AT contains pointers to descriptors of relevant software packages and hardware. These pointers point to the Computer and Software Knowledge Base, described in the next section, which contains implementation and system use details.

The structuring of the AT as a network rather than as a set of hierarchies results from the high level of overlap between many categories of tools and the different backgrounds from which users might seek similar tools. Spectral analysis methods occur in linear systems, statistics, signal processing, and image processing. The additional complexity resulting from using a network rather than a hierarchical representation is, to a degree, mitigated by the fact that the system or the user generally is searching the AT to select a particular tool or alternative set of tools, rather than to select a more extended portion of the knowledge as is done in the formation of an entire graph constituting a plan using the SIKB.

The data presentation tools in the AT will reflect the considerable growth in flexibility of clarity of data presentation made possible by the advent of high resolution screens and windows packages. The tools will include a variety of plotting routines with multiple window output to allow the scientist to compare the results of different analyses and to allow the user to simultaneously examine output at several scales of detail.

Computer and Software Knowledge Base

The Computer and Software Knowledge Base (CSKB) contains information about the software routines available on the computer network, hardware characteristics of the machines on which these routines run, and characteristics of the network. The CSKB provides the information required to automatically form a set of computer commands to carry out the operations required to answer a scientific query, thus freeing the user of the responsibility of handling the details of the software and hardware environments in which the different routines run.

There are many types of software knowledge which must be represented in the CSKB. Clearly, there is a need to represent the basic function of the routines and a description of the input

and output formats. In addition, there may be hardware requirements. For example, it may be necessary for the user to be sitting at a particular type of workstation to run a particular image analysis program, in order to obtain image displays. The CSKB should also contain information about the characteristics of the data to be processed, such as what normalization of the data may be required.

The CSKB must contain information about the computers on the network. Among the types of information to be included are the workload on the system, protocols for communicating with the computer, and available memory and disk storage for the computer. Since some of this information is dynamic, the CSKB should contain procedures to seek the information it needs when it is invoked.

User Knowledge Base

The User Knowledge Base (UKB) contains information about different classes of system users. These classes are defined on the basis of the user's familiarity with the system and with the various scientific disciplines involved in answering various scientific queries. The information in the UKB is used to control the information which is presented to the system user in performing a scientific analysis. There is little use in presenting a user lacking knowledge in a subject with the option of modifying default parameters for a task related to that subject. For example, a user may be performing pixel classification and the system decides that several closely related classification algorithms are likely to be relevant. The system may decide to select one algorithm for the user without explicitly listing the other options, if the user specifies that he has a limited knowledge of classification algorithms. If the user is dissatisfied with the results, the system could inform him of the presence of other options.

The user has access to a tremendous amount of information in the various knowledge bases. The amount of information can be intimidating and it is important to have tools to simplify the user's interaction with the system. In addition to the use of the UKB, the system will use templates with defaults, if desired, to present the user with an example of how to structure any phase of the analysis. For example, the user may have a drainage query. The system will create a partial plan for answering this query and present it to the user as a template in which the user can change various goals and subgoals in the template as desired. Graphical interfaces will be heavily used. The system will be window based and a mouse will be extensively used to allow selection of options. Browsing facilities will be provided for examining the various knowledge bases. The browsing facilities will include visual display of the knowledge bases in terms of graphs and trees to aid the user in understanding the structural relationships represented in the knowledge bases.

The user can customize the system for his use in several ways. First, the user can select the level of detail provided by the system in soliciting responses from the user. Second, the user can define and use names representing frequently used strings or whole sequences of system commands to simplify communications with the system. The user can also define new graphic icons and graphics displays to represent information supplied by the system.

Data Catalog

The data archive catalogs provide on-line summarized information about the data sets which may be off-line. The minimum information contained in the catalog include type of data (e.g., multispectral), platform, time periods active, sensor status history, flight or orbit parameters, other relevant characteristics of the data (e.g., spectral bands of a multispectral scanner), and the location of the data. If summary information is available it would be included. For example, percentage of cloud cover could be stored for multispectral remote sensing data. Several types of cross references are provided, such as references to related data sets, relevant analysis plans, and history of analysis. Note that some of this information is actually stored elsewhere and only a pointer is actually stored in the Data Catalog. For example, sensor status history is stored in the Instrumentation Hierarchy, and only a pointer to the appropriate frame and slot in the Instrumentation Hierarchy is stored in the Data Catalog.

PLAN GENERATION

A flexible and user-friendly system for specifying complex tasks would be of great importance in creating and maintaining large heterogeneous scientific databases. Among the tasks to which a system might be applied are (1) intelligent archiving of data, (2) data retrieval, and (3) data analysis. Since the range of operations required for these tasks is broad and each task may be accomplished using many combinations of more primitive procedures with numerous parameters to be specified, it is critical that the task specification procedure provide the user with considerable guidance in moving through this maze of possibilities. In this section, we describe a general approach to plan generation. While the primary use of this approach is in generating plans for the Scientific Inquiry Knowledge Base, the algorithm is general and could be applied for other plan formation applications.

A key idea in controlling the complexity the user must face in specifying complex tasks is to provide the user with tools to facilitate top-down task design with constraint satisfaction. The user should have the flexibility of freely combining compatible operations to form complex tasks. The system should present, when requested, suggestions to the user on what to do next, based on information about the task to be

performed and the user's background.

We define a task in terms of a goal structure, consisting of high-level goals, and subgoals refining the high-level goals. This refinement can go as many levels deep as necessary. The bottom-level goals should be realizable in terms of existing software and available data. We define a procedure to be a computer routine, together with a description of its input and output. To achieve a goal, a goal structure is required, together with procedures for achieving each of the bottom-level goals. We refer to this combination of goals and procedures as a plan.

A goal is represented by a frame. The frame consists of slots containing information about the function to be performed, characteristics of the data to be processed and other relevant information. A generic goal structure might have the structure shown in Figure 4(a). A goal structure is a tree in which each node is a goal. There will generally be constraints on the order in which the system attempts to satisfy goals. This can be specified by giving a partial ordering on the set of all goal nodes in the goal structure, where the partial ordering represents constraints on the order in which the goals should be satisfied. This partial ordering is implicitly defined by the goal slots in Figure 4(a) called "Input types and conditions" and "Output types and conditions."

A procedure is represented by a frame. This frame gives information about the software used to perform the operation specified by the procedure. An example of a generic procedure structure is given in Figure 4(b). A procedure provides the information needed to run a sequence of computer programs.

To form a plan a user first specifies a goal for the plan. The system then provides, if possible, subgoals which can be used to achieve this goal. The user can replace any or all of these subgoals with new subgoals of the user's design, or accept the subgoals of the system. The user can proceed in a top-down fashion refining the goals in any order. The system will provide the user with possible subgoals at any level of depth in the plan development. The system will also keep track of all constraints contained in the subgoals and guarantee that the set of subgoals specified by the partial plan satisfy all constraints.

The formation of a plan involves integrating knowledge from several knowledge bases including the AT, and the CSKB to produce a plan in the SIKB. The system will automatically go to the relevant knowledge bases to provide choices for the user. Once a plan has been formed, the system evaluates the plan with respect to several characteristics including the likely success of the approach, and the likely computer costs. The user can then request the system to optimize the plan by trying alternative choices for various options. Initially the optimization will be done by considering various choices for the

Goal structure

- Previous applications
- Reliability of results
- Past results
- Relevant techniques
- Computational resources required
- Input types and conditions
- Output types and conditions
- Connections to subgoals

(a) Generic Goal Structure

Procedure structure

- An application routine
- Input types, formats, and conditions
- Output types, formats, and conditions
- Connections to other methods

(b) Generic Procedure Structure

Figure 4. Generic Goal and Procedure Structures for Plans

procedures, since modification of goals may significantly change the nature of the analysis. As experience is gained with the effects of optimization, more extensive structural changes in plans will be considered in the optimization process.

INITIAL IMPLEMENTATION

We are currently implementing a demonstration system on a Sun 3 workstation. CLIPS [1987], which was developed by NASA, is serving as the expert system development tool. The initial system will emphasize the data cataloging and archiving part of the system. The initial applications will be limited to simple but representative data screening type of operations in several domains. We presently plan to use Landsat as the initial sensor to be modelled, and plan to implement a cloud cover analysis and vegetative index analysis program for the demonstration system.

SUMMARY

This paper describes a design for an intelligent ground-based data management system. The system is designed to help automate data cataloging and provide the scientific user with quick access to relevant data and analysis procedures, without requiring the user to have an extensive knowledge of the system. The large volume of data on current and future NASA missions makes such a data management system essential if the data is to be effectively used by a large community of scientists.

The system contains a general artificial intelligence based plan formation capability which can be used to rapidly generate plans for data cataloging and analysis. This plan formation capability provides a hierarchical approach to plan formation. Once a plan is formed the system optimizes it to make optimum use of computer facilities and minimize user waiting time.

REFERENCES

CLIPS Reference Manual, Version 4.1, Artificial Intelligence Section, NASA/Johnson Space Center, September 1987.

N. Nilsson, Principles of Artificial Intelligence, Tioga Publishing Co., 1980.

Modeling and Simulation

**Automatic Mathematical Modeling For Real Time
Simulation System**

**The Space Station Assembly Phase: System Design
Trade-offs For The Flight Telerobotic Servicer**

**A Simulation Engine - Combining An Expert System
With A Simulation Engine**

PRECEDING PAGE BLANK NOT FILMED

AUTOMATIC MATHEMATICAL MODELING
for
REAL TIME SIMULATION SYSTEM

Caroline Wang
Steve Purinton

Software and Data Management Division
Information and Electronic Systems Laboratory
Science and Engineering Directorate
Marshall Space Flight Center/ NASA
Huntsville, Alabama

ABSTRACT

This paper describes a methodology for Automatic Mathematical modeling and generating simulation models. The models will be verified by running in an test environment using standard profiles with the results compared against known results. The major objective is to create a user friendly environment for engineers to design, maintain and verify their model and also automatically convert the Mathematical model into conventional code for conventional computation.

A demonstration program was designed for modeling the Space Shuttle Main Engine Simulation. It is written in LISP and MACSYMA and runs on a Symbolics 3670 Lisp Machine. The program provides a very friendly and well organized environment for engineers to build a knowledge base for base equations and general information. It contains an initial set of component process elements for the Space Shuttle Main Engine Simulation and a questionnaire that allows the engineer to answer a set of questions to specify a particular model. The system is then able to automatically generate the model and FORTRAN code. The future goal which is under construction is to download the FORTRAN code to VAX/VMS system for conventional computation. The SSME mathematical model will be verified in an test environment and the solution compared with the real data profile.

The use of Artificial Intelligence techniques has shown that the process of the simulation modeling can be simplified.

PRECEDING PAGE BLANK NOT FILMED

INTRODUCTION

Mathematical Modeling for real time Simulation programs is a very complicated process which includes Analysis, Design, and the Generation of complex equations and programs. Generally the model will require several modification before it will match a real system. Historically the modifications have been time consuming and a fertile source of errors.

The use of Artificial Intelligence techniques has shown that this process can be simplified. Some of the AI software tools will allow us to create a user friendly environment for engineers to design, maintain and verify their model and also automatically convert the Mathematical model into a conventional programming language for execution.

PURPOSE

The major objective is to develop and create a very comfortable environment for engineers to design and maintain their model. The automatic Mathematical Modeling automatically generates knowledge base, mathematical equations and conventional program codes. It helps to simplify the process of the modeling, cuts down the development time and errors.

The automatic generation of a math model in simulations will add confidence in the simulation. Using interfaces which are well defined and validated will reduce programming and debug time and will allow concentration on logic and equations.

This technique can be applied to many different science and engineering projects.

INTELLIGENT INTERFACES

The intelligent interfaces include:

- (1) The user interface
- (2) The interface between knowledge base and automatic model generation.
- (3) The interface between modules comprising the simulation.

The Automatic Modeling requires a very friendly work environment to collect the necessary information and generate the

knowledge base. Through the knowledge base information, the system automatically creates the mathematical equations and generic program codes for equations. Next, all the equation codes were generated and linked to another AI program to combine and organize the codes together and build a complete compiled program.

An intermodule interface will become part of the automatic model generation process. Variables will be identified and an external interface will provide the variable and its type to external modules. These external modules will record, display or modify the variable.

AUTOMATIC MODELING

The traditional way for simulation modeling is to define, derive and organize the equations and then develop the program for computation all manually. If any modification or correction need to be done in the design or mathematical equations, it will create a tremendous amount of work for the rest of the process. Generally the model will require several modifications before it match a real system. Historically the modifications have been time consuming and fertile source of error.

Some of the available Symbolic mathematics tools will help us to derive the equation symbolically and automatically produce the final equation and program code. The use of Lisp language can build the friendly user interface and generate the knowledge base for the symbolic mathematics tool to build equations and program codes. After the equations and codes have been stored in the files, the Lisp program can then combine and organize them and create a complete compiled source code for any conventional language you required.

The use of Artificial Intelligence techniques has shown that the process of simulation modeling can be simplified.

Control, display and ancillary systems will be developed which will allow the execution of a predefined profile (from a file) or interactive modification of the variables. Variables can be displayed in a text format or as a plot from the control and display system. An execution control module will be available to allow time and sequence control of the model and peripheral models. A recording module will be available for execution in series with the math model and will record (or not record) interface variables transparently. This module will use packet definition variables and circular queues to determine what and when to record.

A. FLOW DIAGRAM

START

i	i
i Define the problem	i
i	i

i	i
i Define generic equations	i
i	i

i	i
i Lisp program for user interface	i
i	i

i	i
i automatic knowledge base	i
i generation	i
i	i

i	i
i Read knowledge Base and	i
i automatically generate the	i
i Macsyma code for Symbolic	i
i Math. tool "Macsyma" to	i
i recognize the information	i
i	i

i	i
i Get in to Macsyma window to	i
i process the set of Macsyma	i
i code which has been generated	i
i and automatically create	i
i the equations and FORTRAN code	i
i for the model	i

i	i
i Save the final symbolic equation	i
i and FORTRAN equation code into	i
i the disk files	i
<hr/>	
i	i
i Another set of Lisp function	i
i to read the equation code and	i
i build a compiled conventional	i
i program for computation	i
<hr/>	
i	i
i Verifying the result and maintain	i
i the equations or knowledge base	i
i and process the whole cycle again	i
i until correct	i
<hr/>	

B. BASIC SOFTWARE and HARDWARE REQUIREMENTS

The Basic software tool we used are "Macsyma" and "Lisp". The Hardware are "Symbolics 3670" and "VAX", we use a Symbolics for the equation and conventional code generation and then down load to VAX system for conventional computation.

There are a number of tools available for Symbolic Mathematical equation and conventional program code generation. Such as:

Symbolics Math. tool	Memory requirements	Available computers
MACSYMA	2 Megabytes	DEC 10 or 20 VAX Honeywell 6000 Series Symbolics LM2 or 3600
SMP	2 Megabytes	VAX 730, 750 or 780 Using UNIX operating system
REDUCE	350 kilobytes	IBM-360 or 370 DEC 10 or 20 VAX UNIVAC 1100 CDC CYBER CRAY-1

BURROUGHS 6700
APOLLO DOMAIN AND OTHERS

MuMATH	512 kilobytes	Personal Computer
MAPLE	350 kilobytes	VAX Using UNIX 4.3
ALTRAN	270 kilobytes	1966 Standard Fortran
FORMAC	150 kilobytes	IBM-360 or compatible machine
SAC-2	120 kilobytes	1966 Standard Fortran

Hardware requirements are for the support of tasking and for the provision of adequate address space.

MAINTAINING AND VERIFYING

One of the most difficult problems in software today is the verification and maintenance of existing programs, especially programs built up over time with many programmers involved.

This is particularly true for simulation programs. The traditional way of modifying simulation programs by rebuilding the model and recoding the model had a potential source of many errors.

The automatic system can eliminate most of these problems. If in error, the only place which is considered to change is the knowledge base at front, then the rest of equation and program can automatically be changed. The procedure becomes much simpler, which makes it easy for the user to maintain and modify the model and program directly.

Improving maintenance and verification is a major goal for this type of model generation. Allowing maintenance to be concerned with equations and logic rather than the implementation is a goal for maintenance.

GENERAL APPLICATION

The Artificial Intelligence technique for Automatic Mathematical Modeling can apply to problem which requires to solve:

- (1) complicated mathematical equation derivation that is hard or impossible for human to do it manually.
- (2) equations which share the generic theory.

or

(3) models which need to be changed a lot.

Robotics, contact dynamics and other problems which require a lot of complicated and long equation derivation in Matrix Multiplication can be solved symbolically very easily.

The Space Shuttle Main Engine Simulation contains a lot of equations where there are sets of equations sharing the generic theory. The Engine model is required to change when the Engine design in hardware is modified. This is the case we used for Automatic Modeling Example.

There are many simulation models which have the need to be changed under different conditions. If they are done manually, it will cost tremendous amount of time to modify the model and change the program code and verify the result before it can be used. The AI technique totally improves the environment and the only place which needs to be modified is the knowledge base, the rest of the product can be automatically generated.

Spacelab training applications where several physical models will be developed for a single flight. Automatic generation of individual models from equations and logic will eliminate much of the repetitive system integration.

EXAMPLE APPLICATION

A demonstration program was designed for modeling the Space Shuttle Main Engine Simulation Mathematic Model called Propulsion system Automatic Modeling (PSAM). Psam is written in LISP and MACSYMA and runs on a Symbolics 3670 LISP machine.

The design goals for PSAM were to develop automatic modeling skills for propulsion system, and other scientific and Engineering applications. We used the old Engine Model for an example to study.

PSAM includes the following features:

- (1). User friendly interface.
- (2). Automatic Knowledge Base generation.
- (3). Automatic Equation and Coding generation.

The Space Shuttle Main Engine Simulation model was built up from the component process elements and their combination into the subprograms.

The component process elements are Pump, Hot gas turbine, Hydraulic turbine, Turbopump, Combustor, Valve, Incompressible propellant flow, Injector Volume with priming for start, Hot gas heat transfer and Regen cooling flow. The subprograms are Fuel, Oxidizer and Hot gas.

There are two types of information for a PSAM knowledge base. One is the component process elements generic equations and the other is the information base for the combination of the Space Shuttle Main Engine model subprograms and component process elements.

The system collects the detailed requirement and generates the set of specific equations for the component process elements and subprograms.

PSAM has the ability to:

- (1). Create or maintain the Knowledge base
- (2). Load different knowledge
- (3). Automatically generate Equations
- (4). Output generated Equation or FORTRAN code to Disk file or option for print out of the Laser Printer.

The Fortran code is in generic conventional program format only for the equations. Another part of the Lisp program will combine all the subprograms and component process elements equations and append the header program codes to become a whole compiled program. The coding format for the equations are the same for many available conventional software such as FORTRAN, C OR ADA ...etc. So This final program can be build on whichever conventional software the user requires by only changing the header requirements and I/O functions.

Several versions and configurations of SSME will be generated and run with a version of the engine controller software.

FUTURE PLANS

Currently we are using the existing project to test the method. In the future we can apply this technique into a lot of new projects. The goal is to build the simulation models and maintain it all automatically.

The knowledge base also will contain the detailed information for automatic documentation.

The future plans for PSAM are:

Generate models based upon logic and flow rather than equations. Build control and display modules which will complement the models.

THE SPACE STATION ASSEMBLY PHASE: SYSTEM DESIGN TRADE-OFFS
FOR THE FLIGHT TELEROBOTIC SERVICER

Accepted for Presentation at the
1988 Goddard Conference on Space Applications of
Artificial Intelligence (AI)
NASA Goddard Space Flight Center
Greenbelt, Maryland

May 24, 1988

Jeffrey H. Smith
Max Gyamfi
Kent Volkmer
Wayne Zimmerman

Jet Propulsion Laboratory
4800 Oak Grove Drive, MS 301-280Q
Pasadena, California 91109

(818)-354-1236

THE SPACE STATION ASSEMBLY PHASE: SYSTEM DESIGN TRADE-OFFS FOR THE FLIGHT TELEROBOTIC SERVICER

Jeffrey H. Smith
Max Gyamfi
Kent Volkmer
Wayne Zimmerman

Jet Propulsion Laboratory
Pasadena, California

ABSTRACT

The efforts of a recent study aimed at identifying key issues and trade-offs associated with using a Flight Telerobotic Servicer (FTS) to aid in Space Station assembly-phase tasks is described. The use of automation and robotic (A&R) technologies for large space systems often involves a substitution of automation capabilities for human EVA or IVA activities. A methodology is presented that incorporates assessment of candidate assembly-phase tasks, telerobotic performance capabilities, development costs, and effects of operational constraints (STS, attached payload, and proximity operations). Changes in the region of cost-effectiveness are examined under a variety of system design assumptions.

A discussion of issues is presented with focus on three roles the FTS might serve: (1) as a research-oriented test bed to learn more about space usage of telerobotics; (2) as a research based test bed having an experimental demonstration orientation with limited assembly and servicing applications; or (3) as an operational system to augment EVA and to aid construction of the Space Station and to reduce the program (schedule) risk by increasing the flexibility of mission operations.

INTRODUCTION

There has been continuing interest in the use of telerobotics for Space Station activities from Congress, the Advanced Technology Advisory Committee, and work package contractors as a possible means for reducing EVA/IVA activities and operations costs, increasing safety, and improving the technology base and spin-off potential of telerobotics (NASA/JSC, January 15, 1987; National Academy of Sciences, 1986). A large-scale analysis of the Space Station assembly phase by the Critical Evaluation Task Force (CETF, 1986) in the fall of 1986 resulted in the accommodation of a Flight Telerobotic Servicer (FTS) as an option for possible use starting at First Element Launch (FEL--the first flight in the Space Station assembly

phase). While the CETF recognized that an FTS could make a substantial contribution to reducing EVA during the assembly phase, it was not clear whether such a system built at a given technical risk would be cost-effective. This question motivated the need for the methodology presented herein.

A key milestone for Space Station assembly, the Permanently Manned Configuration (PMC), is the point at which astronauts can reside for long periods on orbit without returning to earth with the Space Shuttle. The period from FEL to PMC is severely constrained for EVA resources, due to the short (Shuttle-based) time intervals for assembly (approximately one week). There is a need to displace EVA resources where "need" is defined as an FTS capability to reduce crew-EVA time so that absolute Shuttle-based EVA limits are not exceeded. Furthermore, the FTS must accomplish this reduction in a manner that is at least as cost-effective and reliable as available alternatives. The degree of mismatch between task activities and EVA requirements during the assembly phase results in excessive EVA (which is expensive and hazardous), additional power requirements for the Space Station (to support the additional crew to perform the EVA tasks), and additional STS flights to "make up" shortages of EVA time. After PMC, the value of the FTS can be argued to depend on a more complex set of considerations: life-cycle cost, productivity gains, safety improvements, technology spin-offs, and other factors. This paper focuses on cost factors: considerations such as safety and technology spin-off benefits were not explicitly addressed.

The purpose of this paper is to present an approach for assessing the feasibility of utilizing telerobotic systems in the space environment and present the results of an application of the methodology to the Space Station. The results and design issues encountered are based on a recent investigation by the authors [Smith, et al., 1987].

APPROACH FOR COMPARING SPACE STATION TELEROBOTICS OPTIONS

A comparison of Space Station telerobotics options involves many complex factors. The objective is to provide a systems-level methodology that addresses the important components affecting the value of an FTS to the assembly phase. The approach is illustrated in Figure 1.

Technically Feasible Task Set

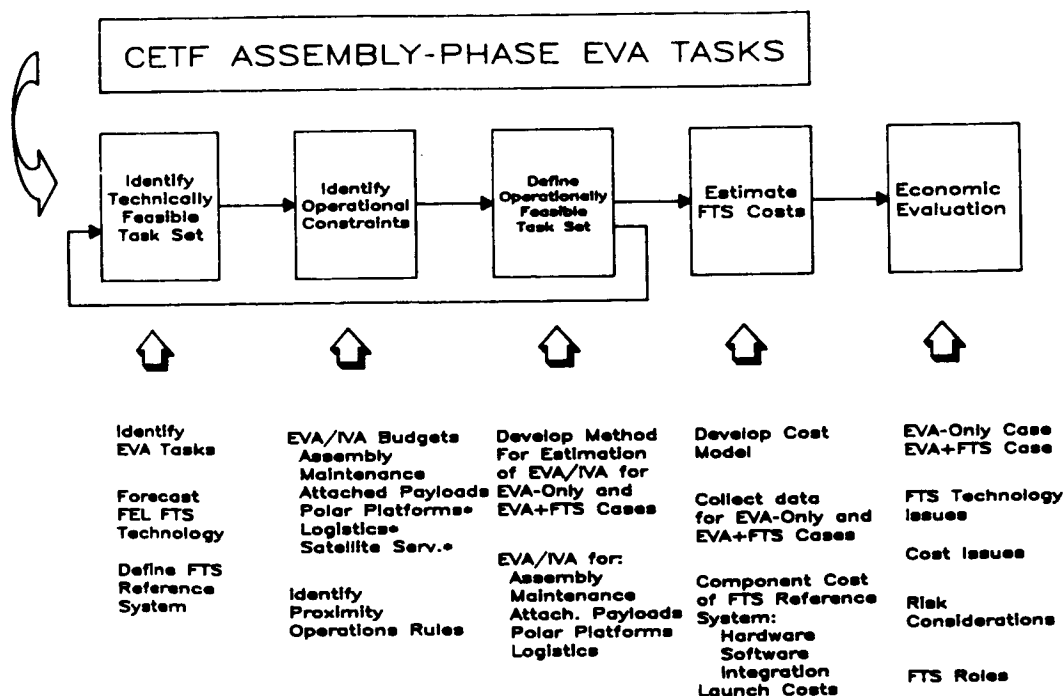
A technically feasible task set is derived from the list of task activities in the areas of assembly, payload servicing, and maintenance. In parallel, an FTS "Reference System" is defined based on a review of potential technologies that will be available by FEL. For the Space Station application, an FTS Reference System is derived that could perform a subset of the assembly phase tasks at a level of technical readiness

corresponding to the FEL date (although the technically feasible task set and corresponding Reference System may initially be somewhat incompatible with total system constraints). However, the purpose of this step is to capture the possible extent of task requirements and capabilities before applying operational constraints to insure the final reference configuration is synchronized with all system constraints.

Operational Constraints

The operational constraints consist of EVA and IVA budgets and proximity operations rules that reduce the technically feasible task set to an operationally feasible task set. The categories of (1) assembly tasks; (2) maintenance tasks; and (3) attached payload setup and servicing tasks; are examined to estimate the EVA and IVA times for two cases: EVA-Only (no FTS) and EVA+FTS (FTS present) (NASA/JSC: March 1986; November 1986; January 8 1987).

The operational constraints are overlaid on the technically feasible task set to derive an operationally feasible task set, and the FTS Reference System definition is revised to reflect the operational constraints. The EVA and IVA times for the two cases were estimated by flight, category (assembly, maintenance, and attached payloads), and year during the assembly phase to measure the savings accrued by the FTS during the operations phase (based on Machell, 1986, McDonnell-Douglas, 1986).



*Examined but not included in the final results

Figure 1. FTS Assembly Phase Study Approach

The FTS Reference System definition is used to generate a bottom-up cost estimate for the economic comparison of the EVA-Only and EVA+FTS cases. The basis for the evaluation is to examine the operational savings due to the FTS Reference System versus the investment cost to design, build, and deliver the FTS Reference System to orbit.

Flight Telerobotic Servicer (FTS) Reference System

To assess the benefits and costs of an FTS, a design concept is required to focus the required technology capabilities and estimate costs. An FTS system is needed that is appropriate for specific EVA tasks required for assembly and operation of the Space Station between FEL and IOC. Such an FTS forecast addresses the availability of critical constituent technologies required at FEL, and highlights essential support characteristics such as FTS reliability, maintenance, and associated logistics support. Selection of technology capabilities must also consider schedule requirements (when must the system be operational), technology and system integration, system verification and testing, and system integration into Space Station operations. The objective is to identify a low-risk, technically feasible FTS Reference System that could be ready by FEL and could perform a set of operationally feasible tasks during the Space Station assembly phase. As the desired functional capabilities are explored, conflicts between FEL functions and technologies are identified and used as discriminators to maintain the list of functional requirements within the realm of feasibility (e.g., tasks requiring a considerable amount of on-line planning for fault management, or a large degree of dexterous manipulation, would not have the commensurate technology in place to meet the task needs) (NASA/JPL, 1986). Tasks considered technically feasible in the FEL to IOC time frame include (1) basic assembly tasks such as pallet handling, worksite preparation, or truss assembly in a well-defined, almost industrial robotic type environment, (2) simple orbital replaceable unit (ORU) change-out and selected inspection type tasks on payloads, (3) Space Station support tasks such as surface cleaning and inspection, (4) pick-and-place type logistic tasks such as transferring components or fluid consumables from the Shuttle to the Station, and (5) other support such as transporting equipment from one place to another, holding equipment in place while it is worked on by EVA astronauts, or providing on-site visual monitoring of an EVA task.

Given a set of possible technically feasible tasks, telerobot technologies are matched against those tasks. The key variables in selecting the technologies are:

- (1) Level of technology readiness (i.e., with FEL being the deadline for delivery)
- (2) Degree of system integration

- (3) Accuracy and repeatability requirements
- (4) Reliability
- (5) Retrofit considerations for future capabilities growth

An important element of technology readiness is whether the technology has the potential for being flight-qualified by FEL (Zimmerman and Marzwell, 1985). Empirical data gathered on system development elapsed time from concept to full operational capability (i.e., space qualification) suggest a time frame between five and ten years for moderately complex systems, and ten to twenty years for complex systems. Therefore, considering the FTS as a moderate-to-complex design with an appropriate logistics support program in place by FEL, it was determined that likely FTS robotic technologies would probably not exceed the present state-of-the-art unless an aggressively funded flight test program or other experience gathering mechanism were introduced to reduce risk.

The next step in identifying a reference system is to develop an array of "strawman" FTS configurations that contain the required robotic technologies while meeting the projected task requirements. For control and vision purposes, the approach is to select the most reasonable reference configuration from the subset of strawman designs. This study, supporting an FEL in the early 1990's, resulted in a reference design having a fixed base in which the fixed base is fastened and the FTS is transported manually to the base using the Shuttle RMS or the MSC where it is connected for operations.

Assembly Phase EVA and IVA Resource Estimates

Due to large uncertainties in some of the data components, ranges are used to bound the results (a formal analysis of these uncertainties was not performed). The total EVA times per flight-interval for the EVA-Only and EVA+FTS cases are illustrated in Figure 2 using low-range EVA estimates for assembly, maintenance, and attached payloads. The low-range values represent the lowest estimates for the EVA range obtained by adding all the low values together. A similar procedure was used for the high-range estimates. The aim was to bound the actual values by examining the extreme low and high values. The estimates of Figure 2 are troubling. The estimated EVA required on five flights prior to PMC exceeds the budgeted amounts of 24 hours. This finding supports the argument that the CETF assembly sequence does not manifest within the CETF constraints for at least three early flights. This is due primarily to assembly on flights 1 and 2 and maintenance and attached payload contributions on subsequent flights. The implication is that for the CETF design to work, one or more shuttle flights must be added, the current shuttle flights must be extended (unlikely), or there must be a re-manifesting of assembly EVA to meet the constraints. It is the cost of additional shuttle flights that dominates the cost-effectiveness of the FTS.

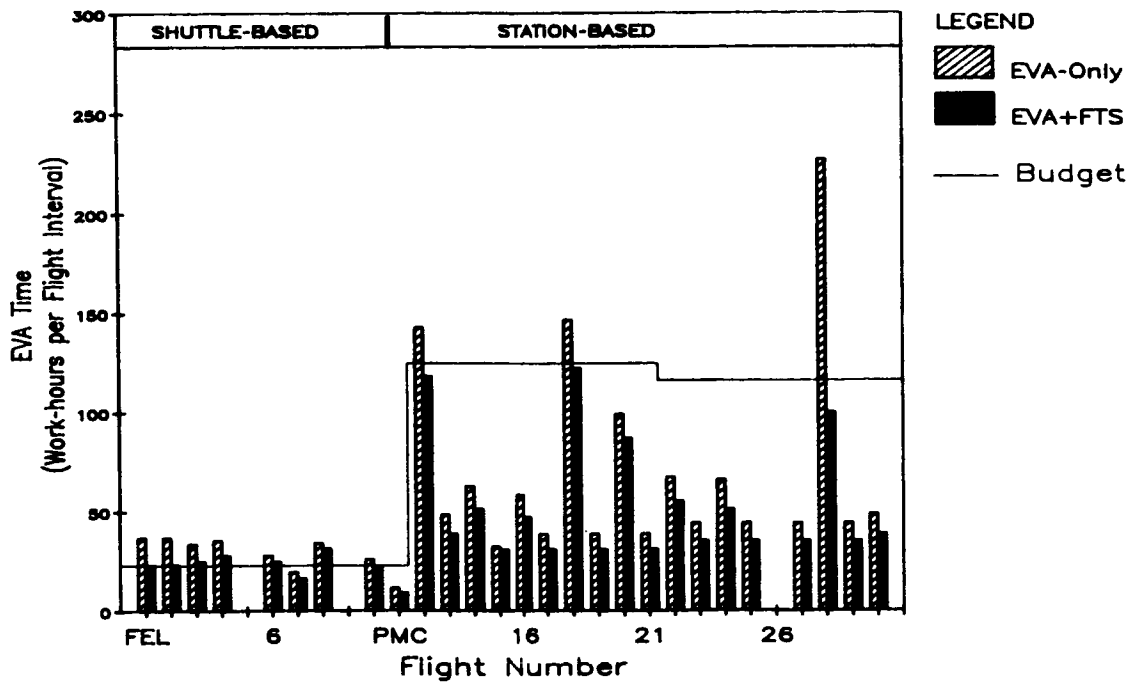


Figure 2. Assembly Phase EVA Estimates--Low-Range EVA Values

RESULTS: ASSEMBLY PHASE COMPARISON WITH AND WITHOUT THE FTS

An economic model was developed to examine the cost-effectiveness of the FTS Reference System and to determine whether the FTS could be cost-effective during the assembly phase. The Net Savings model is:

$$\begin{aligned} \text{Net Savings Due to the FTS Reference System} = & \\ & (\text{Operations and Maintenance Cost of EVA-Only Case} \\ & \text{minus} \\ & \text{Operations and Maintenance Cost of EVA+FTS Case}) \\ & \text{minus Investment Cost of the FTS.} \end{aligned}$$

If the Net Savings is positive, the FTS Reference System is cost-effective. The use of this approach required a cost estimate of the FTS Reference System and a bottom-up cost (component-by-component) estimate was made using the component list for the FTS Reference System [Smith, et. al., 1987]. An estimate of \$277 million (M) to \$304 M was obtained for the FTS (excluding non-prime costs--the costs of managing the prime contracts and spares costs). The costs and benefits of FTS development through completion of the assembly phase were examined. At issue was the feasibility of using the FTS to assist in the assembly process. Thus, benefits to users or the Station after the assembly phase were not examined. FTS ground operations costs were included using estimates of FTS operating costs. Using these cost

estimates and EVA and IVA profiles, a sensitivity analysis was performed to observe the effects on the feasible region.

The results indicate that a key trade-off is between the cost of the FTS and the cost-per-flight of the STS. Because of cases where the estimated EVA exceeds the budget of 24 hours during FEL to PMC, additional flights must be added to make up the difference. The cost of any added flights is a major factor in the cost-effectiveness of the FTS. Figure 3 presents one such trade-off region using the low-range estimates of EVA/IVA and the FTS cost over a range of STS costs per flight from \$105M to \$178M. It is difficult to determine an estimate for STS prices. Estimates have ranged from below \$100M to \$150M during the pre-Challenger era. The assumption was made that the price will be higher in the post-Challenger era due to increased safety and reliability requirements, component re-designs, and quality control constraints. However, a range of price curves is presented to provide a generalized result. The FTS cost ranges from a low \$232M (NASA estimate) to \$340M (National Research Council, 1987); the end points were selected merely to limit the scope of the trade-off region. The area in the center of the region bounds the FTS Reference System estimated costs. As an example, if we assume a STS cost of \$150M, the FTS will break even if it can be built for a cost of \$292M or less. If the FTS costs more than \$292M, it will not be cost-effective (unless the STS price is actually higher). For the other points on any of these curves, the estimated net savings can be read from the axis on the left.

Also note the term "Mixed Manifesting" on Figure 3. This refers to assumptions made regarding how excess EVA is re-manifested on subsequent flights if an additional flight is required. For the mixed manifesting case, if EVA is required on the early flights (1-5), manifesting is inflexible and after Flight 5, a flexible scenario is assumed. Note that as manifesting becomes inflexible, the FTS cost effectiveness region moves up (toward cost-effective) and as manifesting becomes flexible, the FTS cost-effectiveness moves down (toward less cost-effective).

If the scenario is moved toward the flexible manifesting assumption, the trade-off region moves down (toward less cost-effective) because fewer overall flights are required. If the scenario is moved toward the inflexible manifesting assumption, the region moves up (more STS flights are required). Furthermore, as the difference between the number of additional flights in the EVA-Only case and the EVA+FTS cases (if any) becomes larger, the width or spacing between the curves also becomes larger. The constant slope of the curves (approximately -0.75) is an indication that for each reduction in FTS cost of one dollar, there is an increase in net savings of only \$0.75. The remaining 25% is the delivery cost and the effects of discounting.

FTS VS. STS TRADE-OFF REGION Low EVA Estimates/Mixed Manifesting

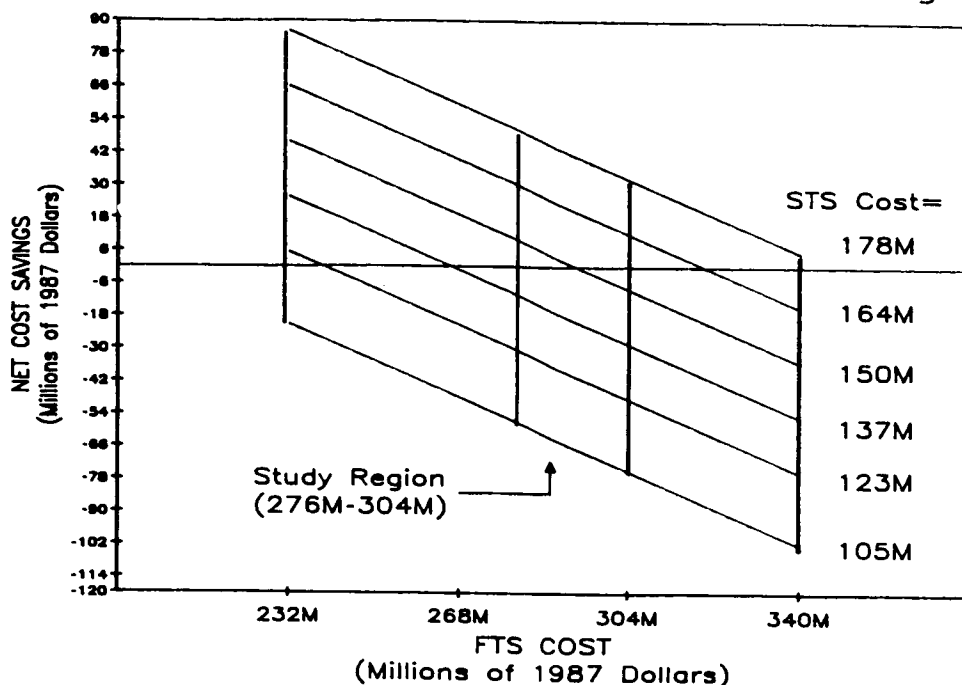


Figure 3. FTS versus STS Trade-off Region

The effect on the feasible region of moving from low-range to high-range EVA values is shown in Figure 4 (the region moves down). Similarly, as the estimated cost of the FTS increases, cost-effectiveness drops (the region shifts downward).

Another parameter of interest is the EVA cost per hour used to estimate the cost of EVA hours used. As with the STS cost, the estimation of such a value is difficult. To examine the sensitivity of the results to EVA cost per hour, three cases were examined using \$45,000 (\$45K), \$35K, and \$25K per hour (Figure 5). Note the apparent insensitivity of the region to this parameter. This is due to the magnitudes of the numbers between the FTS and STS costs. A decrease in the cost per hour simply places less value on the resource benefits the FTS can displace and thus makes the FTS region move down. The discount rate used in the above results is the Office of Management and Budget (OMB) value of 10% used for cost-benefit analysis on government projects. The effect of varying the discount rate was also examined using a 6% rate. The effect of reducing the discount rate is to move the trade-off region up significantly (Figure 6). This indicates that a lower discount rate would have a significant impact on improving the cost-effectiveness of the FTS.

FTS VS. STS VS. EVA RANGE High/Low EVA Estimates/Mixed Manifesting

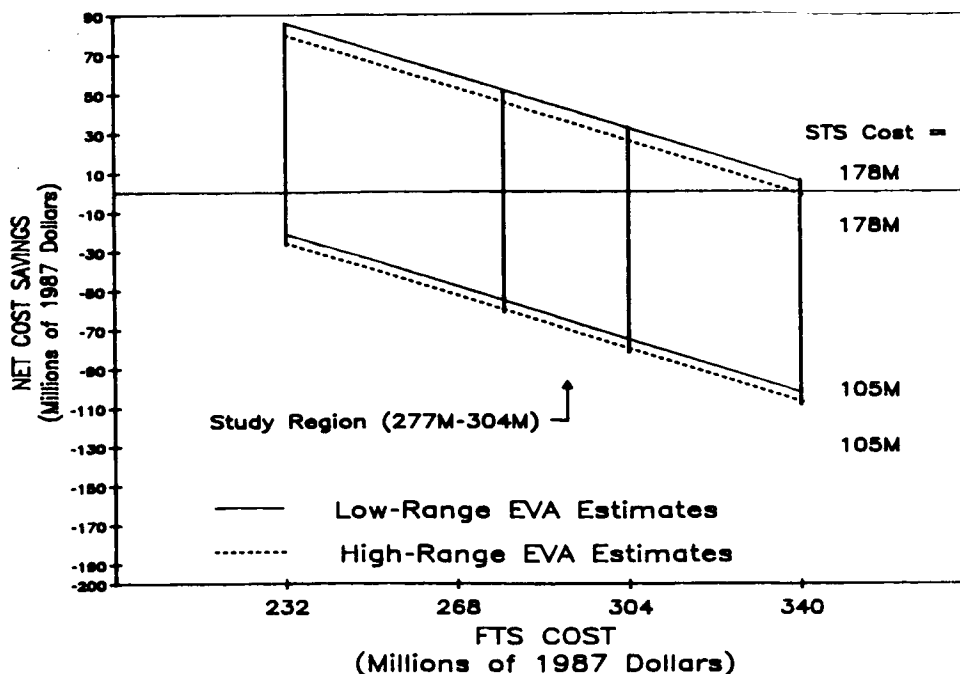


Figure 4. Low-Range versus High-Range EVA Cost
FTS VS. STS VS. EVA COST/HR.
Low EVA Estimates/Mixed Manifesting

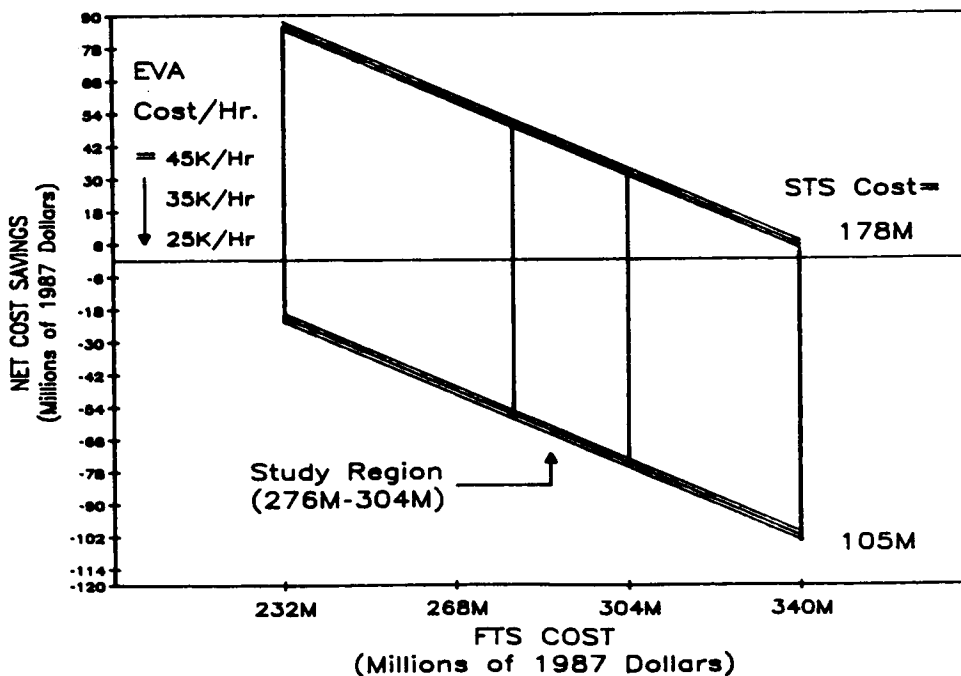


Figure 5. Sensitivity to EVA Cost Per Hour

FTS VS. STS AT 6% DISCOUNT RATE Low EVA Estimates/Mixed Manifesting

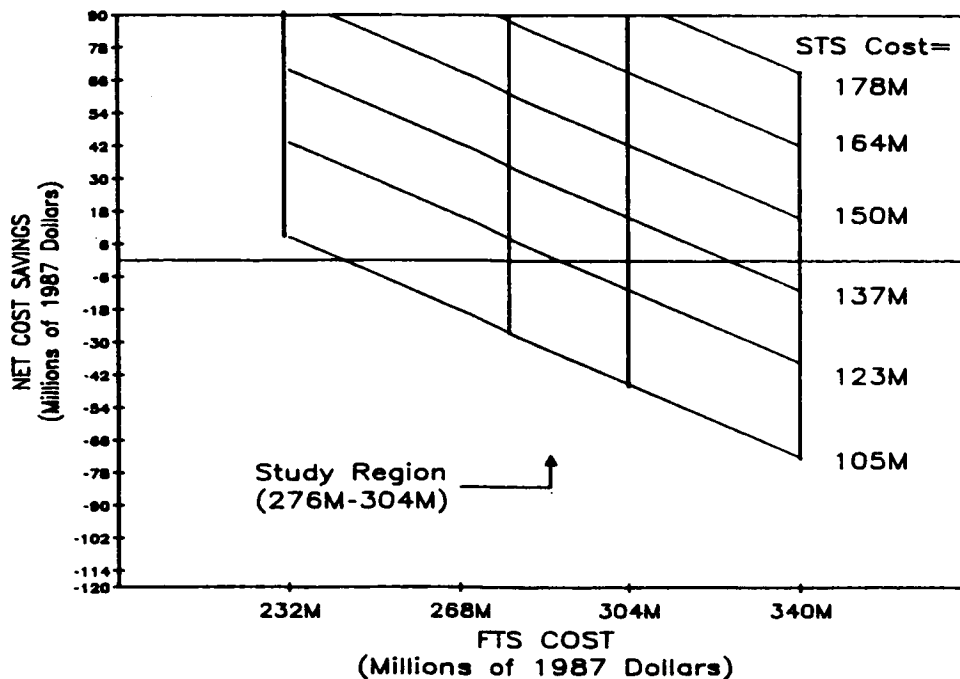


Figure 6. Feasible Region at a Lower Discount Rate

DESIGN ISSUES AND IMPLICATIONS

A key design issue is to identify the goal of the overall program that affects the trade-offs to be made between the numerous users the robotic system faces. If the value to be maximized is the commercial benefit to be derived from technology advances (i.e., spin-off potential), then a different value equation (than net savings) will need to be constructed in order to accommodate those technologies to be stimulated, and thus the activities that the FTS can be used to demonstrate. It was assumed here that the objective was to maximize the overall value of the FTS to the Station. Thus, technology development programs need to be instituted that enable FTS performance upgrades in areas that directly enhance FTS value to the Station. This could be done by identifying high-payoff applications amenable to acceptable-risk FTS system configurations. This assumption need not minimize the role of the FTS program in stimulating automation and robotic (A&R) technology development since both terrestrial spin-off and Station benefits can accrue from development of intelligently selected advanced technologies.

The current study was performed over a period of time in which the Station design moved from the CETF concept to an abbreviated Phase I configuration. However, because the STS-based EVA activity is still highly constrained in the Phase I case, the results are likely to be robust. This is conjecture and should be verified by performing the additional analysis.

It is important to note that whether or not the FTS is cost-effective for the assembly phase, there are legitimate uses under a number of scenarios. If the FTS is not cost-effective, it could still serve as a research and development test bed for post-IOC applications. If it is cost-effective, it could be used as an applications-oriented tool. Earlier studies have highlighted some of these role differences varying from a low-cost orbiter-based operational system to a space-based test bed for evolving telerobotics technologies [Goddard, 1986]. Although there is a range between an applications-oriented versus a demonstration-oriented FTS, even if marginally cost-effective, the FTS could still serve as a backup that could reduce schedule risks by providing a flexible option for some additional EVA activity if needed. This is an important design issue because it must be shown that a net risk reduction exists. Situations where the added risk of a large robot system that could fail into a dangerous mode or require extensive maintenance or EVA attention must be understood prior to dedication of the system to an operational role. A robotic system can play a testbed or demonstration role in order to gather experience with on-orbit operations at a point where the design of the operational system can be modified. The interfaces between the human operators, the equipment, and the task requirements can be refined or revised to make better use of the synergistic potential of re-designed tasks coupled with FTS capabilities specifically designed for those tasks. If it is assumed that FTS operations are terminated at IOC, or that the FTS is not used for Station operations but rather for research and demonstration purposes, then there are other benefits this paper made no attempt to quantify. One class of benefits is the development of "lessons learned" that can be utilized to develop a future FTS that does play an integral role in a wider variety of Station and on-orbit operations. Such experiences would provide a valuable database for guiding the design of future tasks and FTS capabilities.

Note that the analysis performed herein is inherently conservative. Limiting the time frame of the analysis to the assembly phase underestimates the actual benefits of an FTS by excluding any post-IOC benefits. If the FTS is assumed to continue operations after IOC, the FTS feasible region will tend to move upward (towards more feasible) for all cases. This paper presents a single solution out of many possible ones, and the results described are by no means optimal. The FTS option selected here was based on an analysis of estimated task requirements and estimated functional requirements. The focus was to identify components that ought to be examined when comparing FTS options. Nonetheless, a number of recommendations are made.

There is a need to examine the effects of risk in these comparisons (Smith, et. al, , 1985). Cost risk can be viewed directly using the net savings or operations and maintenance

(O&M) equations with simulation techniques to generate probability estimates for net savings and O&M costs. Then, as assumptions of the problem (such as software/integration costs) are varied, the impact on the probability of breaking even can be computed. Technical risk can also be studied in terms of the uncertainties in performance and reliability. In addition, the effects of specific risk elements, such as the effects of introducing suits requiring no pre-breathe step, EVA overhead, and the effects on EVA if such a suit is not ready on schedule, could be singled out. An understanding of the risk and uncertainty effects would show how the FTS could help reduce program risk by adding flexibility to operations planning and contingency planning--especially during FEL-PMC. There is value and benefit of having an FTS for the flexibility it provides for dealing with unscheduled events. Further study of the risk elements would quantify those benefits.

Additional study is also needed for the allocation of automation and robotic functions. Very different results can be achieved by locating such functions on the ground. With improved autonomous operations, Station IVA could be reduced. One question is whether to pursue advanced and technically risky autonomous or semi-autonomous options versus a less sophisticated on-the-ground remote telerobot operation capability. Such activity would identify the issues related to the human factors and control technology problems of dealing with time delays in teleoperation feedback. It may be possible to mitigate the problems of such time delays with predictive control type technologies. The present paper has shown the magnitudes of the savings to be potentially large enough that a dedicated FTS relay system to provide near real-time response might be an alternative worth considering. This will depend on the potential for extending the displacement of IVA and EVA task times while minimizing the technical risk of developing the system. If extended operations can be performed from the ground, the risk of requiring additional flights may be reduced and provide a schedule margin during the early FEL-PMC period when assembly elements must be completed within fixed, short term flight periods or risk mission failure. The area of allocation of autonomous and robotic functions and resources needs further examination to help designers select whether A&R upgrades are performed on the Station, incorporated into the FTS, or operated on the ground (see Zimmerman, et. al., 1985).

A related allocation problem that requires further understanding is the allocation of work among and between multiple robots (FTS, RMS, MSC, etc.) and crew EVA (co-EVA). Data on performance time ratios for such mixed tasks should be collected for a variety of tasks using neutral buoyancy studies and (eventually) on-orbit experience. Proximity operations rules for such operations will also have to be identified in detail.

DISCUSSION AND CONCLUSIONS

A number of conclusions can be drawn, based on a CETF-derived (30-flight) assembly phase. Noting that the results are conservative in that benefits after the assembly phase are not examined; logistics benefits were not considered; safety benefits were not considered; and the effects of the satellite servicing facility were not examined; the following conclusions were drawn:

- (1) The FTS Reference System identified herein appears to be technically feasible for development by FEL.
- (2) The FTS Reference System is cost-effective under a variety of conservative scenarios.
- (3) The STS cost is the primary factor for FTS cost effectiveness due to avoidance of extra STS flights by EVA reductions.
- (4) The FTS is cost-effective at a 10% OMB discount rate but even more cost-effective at a 6% rate.
- (5) From an EVA resource perspective, the assembly phase is a maintenance problem (50% of total EVA is for maintenance versus 33% for assembly). FEL-PMC is the primary assembly problem.
- (6) The FTS Reference System defined here is most suitable for performing:
 - (a) Truss assembly tasks
 - (b) Limited ORU replacement tasks
 - (c) Deployment of special equipment
 - (d) Pallet handling, loading, and unloading tasksThe potential exists for transferring some on-orbit tasks to ground operations given that appropriate technology and human engineering constraints are considered.
- (7) The total estimated cost of the FTS Reference System is \$277 to \$304M (does not include non-prime costs or spares).
- (8) There is a need for improved and more detailed data on task descriptions, timelines, manifests, etc. updated quarterly or semi-annually and available via electronic mail, for example.
- (9) A methodology for comparing automated and robotic options has been developed with specific applications to the FTS and its technical and cost feasibility for use during assembly phase construction. Other A&R elements could be analyzed in a similar manner.

The approach described in this paper is intended to assist in the characterization of an assembly role for which an early robot or FTS might best be designed. Potential for cost-effective early operation argues for an FTS and host environment designed to facilitate performance of the selected FTS tasks. On the other hand, marginal early operating benefits suggest the

option of treating the FTS initially as a test bed for development of advanced technologies that will later serve the Station in a more cost-effective manner.

A related issue is that of reliability, or more accurately, program confidence in the reliability of the FTS to perform tasks determined analytically to be cost-effective. The Advanced Technology Advisory Committee and Space Station work package contractors have been remarkably consistent in their conclusions regarding which tasks were within the capabilities of telerobotic devices. Program personnel, citing the criticality of early (pre-PMC) EVA tasks, are considerably more skeptical. The CETF, for example, ultimately based its results on the use of deployable utilities in preference to use of an FTS, on the grounds that on-orbit assembly by telerobotic devices had never been attempted. This suggests that the subject of both ground and flight demonstrations of the FTS should be directed specifically toward whatever tasks the FTS might be applied to initially, particularly in cases of high task criticality.

Finally, multiple competing goals have been articulated for the mandated FTS development program and it is not clear that the program adequately addresses this issue. For example, the goal of increased Station productivity and decreased operational cost implies a high-reliability, low-risk, low-maintenance FTS that can be brought on-line early in the Station operating life. This approach cannot be easily reconciled with the current program focus on implementing advanced technologies and system concepts in an operating environment for which no prior operating experience is available. While representing potentially higher technology spin-off value (a separate FTS goal), the technology-driven approach is also of higher risk and possibly of considerably smaller direct value to the Station. Maximizing spin-off value may isolate development attention on technologies that are not particularly applicable to high-payoff Station tasks; also, systems utilizing complex, advanced technologies tend to require larger amounts of maintenance until those systems are mature and well-proven. This could constitute a significant additional burden on Station resources. Finally, any lack of confidence in the reliability of the FTS may cause it to be relegated to "elective" or demonstration functions, rather than being accorded full operational status and assigned to important routine Station tasks.

ACKNOWLEDGMENTS

This paper presents, in part, one phase of research carried out by the Jet Propulsion Laboratory, California Institute of Technology, Space Station Project, which is an agreement under JPL Contract Number NAS 7-918.

REFERENCES

CETF, Critical Evaluation Task Force, NASA Langley Research Center, Langley, Virginia, August 20-September 14, 1986.

Machell, R.M., "Space Station EVA Time Requirements", (McDonnell Douglas Astronautics Company), presented at the Critical Evaluation Task Force (CETF) Meeting, NASA Langley Research Center, Langley, Virginia, August 20-September 14, 1986.

NAS, Report of the Committee on the Space Station of the National Research Council, National Academy of Sciences, National Academy Press, Washington, D.C., September 10, 1987.

NASA/JPL, Functional Requirements for the 1988 Telerobotics Testbed, JPL Document No. D-3693, Jet Propulsion Laboratory, Pasadena, California, October, 1986.

NASA/JSC, Space Station Mission Requirements Data Base, Johnson Space Center, Houston, Texas, March 1986.

NASA/JSC, Space Station Program Definition and Requirements, Section 3: Space Station System Requirements, Revision 3, Document No. JSC 30000, Johnson Space Center, Houston, Texas, November 26, 1986.

NASA/JSC, Space Station Assembly and Maintenance Architectural Control Document, Document No. JSC 30502, Level B Change Request BJ020195, Johnson Space Center, Houston, Texas, January 8, 1987.

NASA/JSC, Flight Telerobotic Servicer Baseline Configuration Document, Document No. JSC 30255, Johnson Space Center, Houston, Texas, January 15, 1987.

Smith, J.H., Feinberg, A., Lee, T.S., Miles, R.F., Reiners, T., and D.L. Schwartz, Autonomy Evaluation Methodology: A Microcomputer Tool for Design Trade-Offs of Spacecraft Systems, Vols. I and II, JPL Document No. D-2761, Jet Propulsion Laboratory, Pasadena, California, November 1985.

Smith, J.H., M. Gyamfi, K. Volkmer, and W. Zimmerman, The Space Station Assembly Phase: Flight Telerobotic Servicer Feasibility, JPL Document No. 87-42, Jet Propulsion Laboratory, Pasadena, California, September 1987.

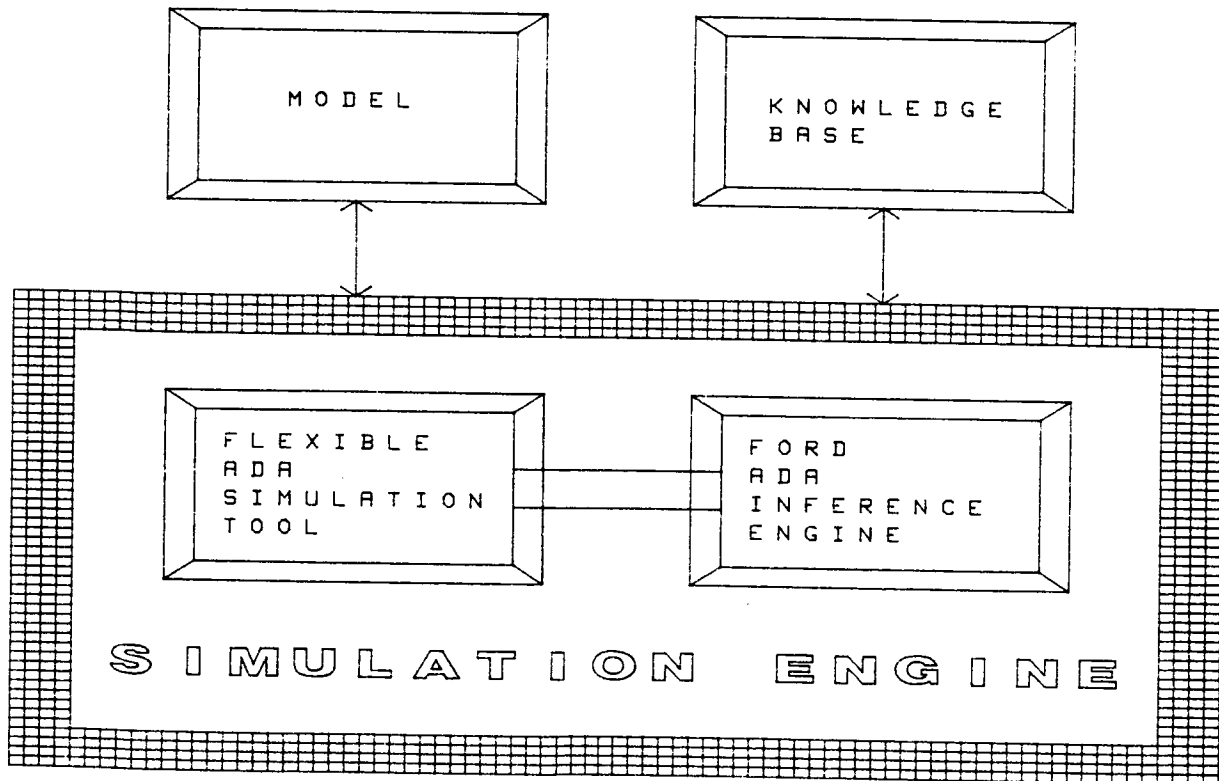
Zimmerman, W.F. and N. Marzwell, "Space Station Level B Automation Technology Forecasting/Planning Structure", JPL White Paper prepared under sponsorship of the Johnson Space Center, Houston, Texas, April 30, 1985.

A SIMULATION ENGINE -
COMBINING AN EXPERT SYSTEM WITH A SIMULATION LANGUAGE

James R. Spiegel and David B. LaVallee
Ford Aerospace Corporation
7401 D Forbes Boulevard
Seabrook, Maryland 20706

ABSTRACT

Expert Systems have been applied in a number of ways to the field of simulation. One of these is the application of an expert system to "drive" a simulation, by making run-time decisions which effect the simulation. This approach has been successful for a number of specific simulation models. The "simulation engine" extends this capability by supporting this type of interaction in a general purpose setting. A general purpose simulation language is provided for building models, and an inference engine is provided for knowledge processing. This combination results in a mechanism which allows general purpose models to be simulated in concert with interactive knowledge bases.



INTRODUCTION

This paper describes an expert system which is fully integrated with a simulation language. This integration was facilitated by the fact that the simulation language was initially designed to be interactive. The integration process was a simple case of teaching the simulation to interact with the expert system.

Classical simulation languages have been developed through time on top of a basic batch oriented foundation. A number of languages have subsequently added an interactive feature to support interactive debugging activities. This paper presents a simulation language which was designed to be interactive. The design objective was to build a simulation environment which is actually driven through the interactive user interface. In this case, the subsequent addition is to substitute an expert system in place of the user.

The system described here provides an application where an expert system is operated in parallel with a simulation. The simulation is actually a user defined model which is written in a "full service" simulation language. The result is symmetric in nature. In addition to the expert system being used to drive a simulation, the simulation may be used to test an expert system. Thus we present an application of simulation to the field of expert systems.

Both the expert system and the discrete event simulation language were independently developed using the Ada programming language. This paper provides descriptions of the expert system (FAIE), the simulation language (FAST), the interface which connects them, and a sample application for which this was utilized.

THE FORD ADA INFERENCE ENGINE

The Ford Ada Inference Engine (FAIE) is an expert system inference engine designed to execute as an Ada task embedded in an expert system, which could in turn be embedded in a larger program. FAIE is an application independent system. It is completely generic in the sense that any knowledge that can be represented in rule format can be encoded, and any program that can interface to Ada can embed FAIE for expert system capabilities.

The knowledge base is represented as a directed acyclic graph. This can be thought of as a network of nodes with the links all pointing in the same direction. The leaf nodes represent initial data points that must be provided to the inference engine. The intermediate nodes represent hypotheses or subgoals that will be tested. FAIE supports the use of AND, OR and NOT constructs. The links between the nodes are the "production rules" that the inference engine uses to traverse the graph. These lead to the eventual firing of the goal nodes on the other side of the graph.

The actual development of the knowledge base is performed using the Knowledge Editor Graphical System (KEGS), which runs on the Symbolics 3640. Figure 1 illustrates a sample knowledge base. The Symbolics is extremely well suited to perform this task. The knowledge base is then downloaded onto a VAX computer, where the inference engine is able to process this knowledge in an efficient manner.

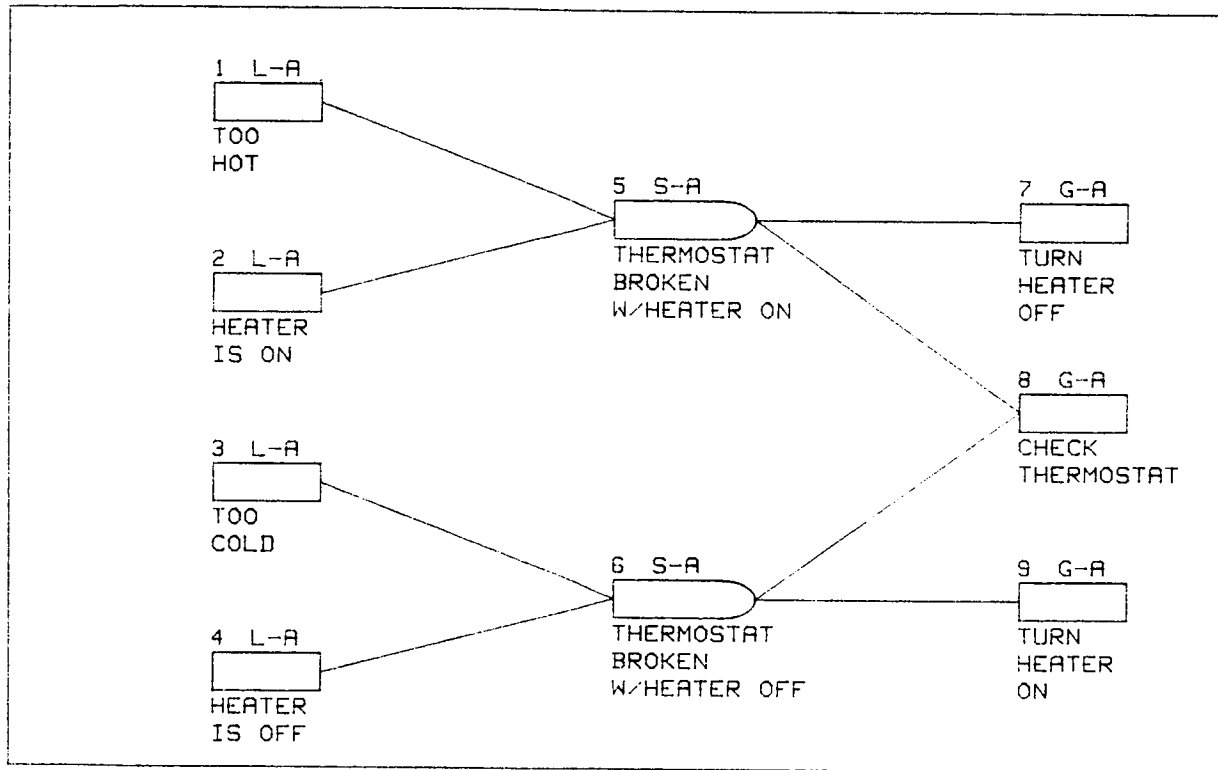


Figure 1 - Sample Knowledge Base

The inference engine uses a combination forward-chaining and backward-chaining algorithm in order to minimize the time spent in proving a goal. In practice FAIE is capable of exceedingly fast performance. On its current VAX 11/780 implementation, rule-firing occurs at the rate of 1500 rules per second. For small expert systems this rate is more than adequate to achieve real-time performance. Moreover, the maximum search time for a goal can easily be computed from the characteristics of the knowledge graph. Future versions of FAIE which take advantage of multiprocessing implementations of Ada run-time systems will be even more powerful.

ORIGINAL PAGE IS
OF POOR QUALITY

THE FLEXIBLE ADA SIMULATION TOOL

The Flexible Ada Simulation Tool (FAST) is the first successful implementation of a discrete event simulation language in Ada. FAST is fully documented and has been developed using the most modern object-oriented software methodologies. The goals driving the development of FAST were to allow a "user-friendly" interactive environment, and to enable enhancing the model with a minimum of knowledge regarding the underlying structure of the simulation tool. The achievement of these goals has resulted in an overall environment which includes both manual and automatic monitor and control of the system being simulated.

FAST is comparable to other general purpose simulation languages such as SLAM, GPSS, SIMAN or TESS, but provides many additional features for dynamic control of simulations, including a full complement of programming language capabilities. FAST is distinguished by its interactive capabilities which allow monitoring and control of simulation in real-time, a facility found in no other existing simulation language. A user may use FAST interactive features to debug a simulation, tune a network, or perform a sensitivity analysis.

The high level FAST design involves four distinct areas: the simulation, monitoring the simulation, controlling the simulation, and the user interface. Figure 2 is a conceptual diagram of their interactions. While a simulation is running, the user is able to monitor one of a large number of pre-defined display pages. The user is also provided with a robust command language which he may use to control the operation of the simulation as well as change model parameters.

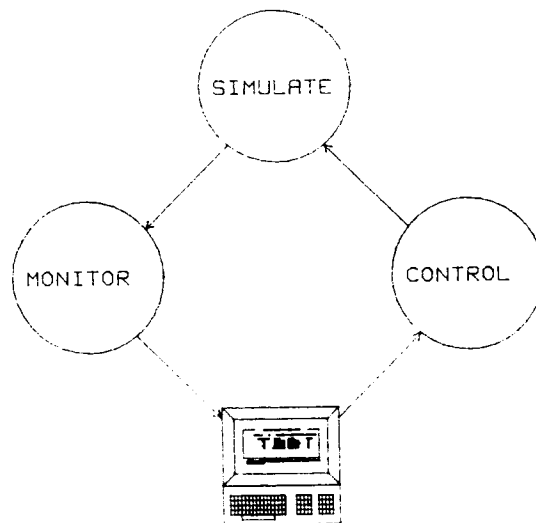


Figure 2 - FAST Simulation Environment

One of the key features of FAST is the ability to interactively monitor and control simulation runs. The interactive interface provides a "control center" environment in which to perform simulations. While a simulation is running, the user has complete visibility into the low level internal data structures as well as into the high level simulation statistics. The result is an interface which provides an extremely powerful debugging capability. In addition, it provides a concept and design that leads naturally to interaction with an expert system.

The display pages which output internal data structures are used to learn exactly what is going on during a run. For example, some form of a "Future Events Queue" is maintained internally by all discrete event simulation systems. FAST allows the user visibility into this structure through the future-events-queue page (Figure 3). While monitoring this page, he may advance the simulation step by step to see what events are being processed, and how they effect the state of the system.

FUTURE EVENTS QUEUE PAGE					
SIZE =	4	CURRENT SIMULATION TIME IS			
		34.7945			
PASSPORT	TIME	VERB	PATH	BOX	
41	34.7945	REQUEST	1	2	
3	35.0045	GENERATE	1	1	
30	35.0837	RELEASE	1	5	
2	250.0000	COMPUTE	2	6	

ERROR : EXECUTION SUSPENDED ERROR : QUEUE OVERFLOW
STATE MENU : ■

Figure 3 - Future Events Queue Page

The command capability also allows the user to change the values of simulation parameters. This idea suggests an operational scenario where a user can monitor a simulation run and adjust simulation parameters to insure that pre-specified system performance requirements are met. For example, if a communications link is being modeled, the user may monitor the

transmission delays. If the delay time gets too long, the user may send a command to increase the link bandwidth. The result is that when the experiment is complete, the user knows what bandwidth is required to meet the specified delay requirement. The next logical step is to replace the user in this loop with an expert system (Figure 4).

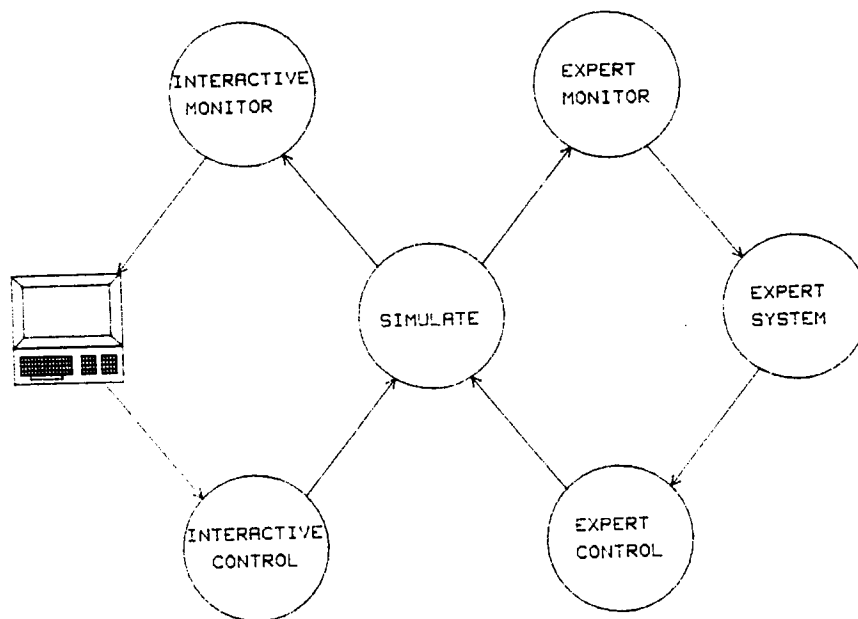


Figure 4 - Integrate Simulation and Expert System

THE FAST/FAIE INTERFACE

In order to have an expert system drive a simulation, it was necessary to provide the monitor and control functions to an expert system. This is accomplished conceptually by extending the user interface of Figure 2 with the expert system, as in Figure 4.

Minor modifications had to be made to FAST and FAIE in order to connect the two. The interface is accomplished through the use of an interface task which has visibility into the data structure of each, and the capability to provide the required translations. For an expert system to work in tandem with the simulation, a means of transmitting specific data to the expert system at a periodic rate was required. FAST currently regularly sends outputs to a screen. In addition, specified outputs are also sent to the interface task, which is able to use this information

to set the values of the leaf nodes. To utilize this feature, the knowledge engineer must decide which data must be monitored by the expert system. He then configures the software to send this information periodically. The front end to the expert system then evaluates the data to see if there are any problems that require action. If so, the appropriate nodes are marked in the knowledge base graph, and the inference engine is activated to solve the problem.

A solution is found when a goal node in the knowledge base is fired. When FAIE fires a goal, this is translated into a pre-determined FAST command. The FAST commands are executed just as if they were typed at an input console. The FAST interface had to be modified to accept commands from this new source, but the format of the commands and their processing all remained the same.

COMBINING FAST AND FAIE

The real-time interaction of an expert system running in parallel with a simulation experiment provides an extremely powerful combination. For example an expert system may be used to "fine tune" the parameters of a particular system being modeled. In this case, the expert is provided with rules which are used to alter the system parameters toward the goal of optimizing a given design objective. Conversely, the simulation capability may be used to test and validate an expert system. For example, if an expert system contains a knowledge base that is designed to perform fault diagnosis for some system, that system may be modeled using the simulation language, and specific faults may be introduced through the simulation real-time user interface. An example of each of these cases is presented here.

USING AN EXPERT SYSTEM TO DRIVE A SIMULATION

When configured with the expert system, the simulation is essentially dual-ported. There is the existing monitor and control function by a user through the terminal. In addition, the embedded expert system monitors specific simulation results and issues FAST commands to alter the simulation. The combination of the simulation run with the expert system provides a real-time solution to what is classically a multi-step, manual process.

The example here deals with sizing storage requirements for a storage system. The model consists of a number of data streams which enter a system and need to be stored on disks. The expert system is used to determine how many disk drives are needed in order to meet system performance requirements. Figure 5 illustrates the knowledge base which is used. The expert inputs are a description of the current performance of the system, in terms of delays and buffer requirements. The expert "goals" are to increase or decrease the number of disk drives. When one of these goals is reached, the interface process will send a command

to the simulation which implements this action. When the run is complete, the system parameters which are needed in order to meet the system performance requirements will have been found.

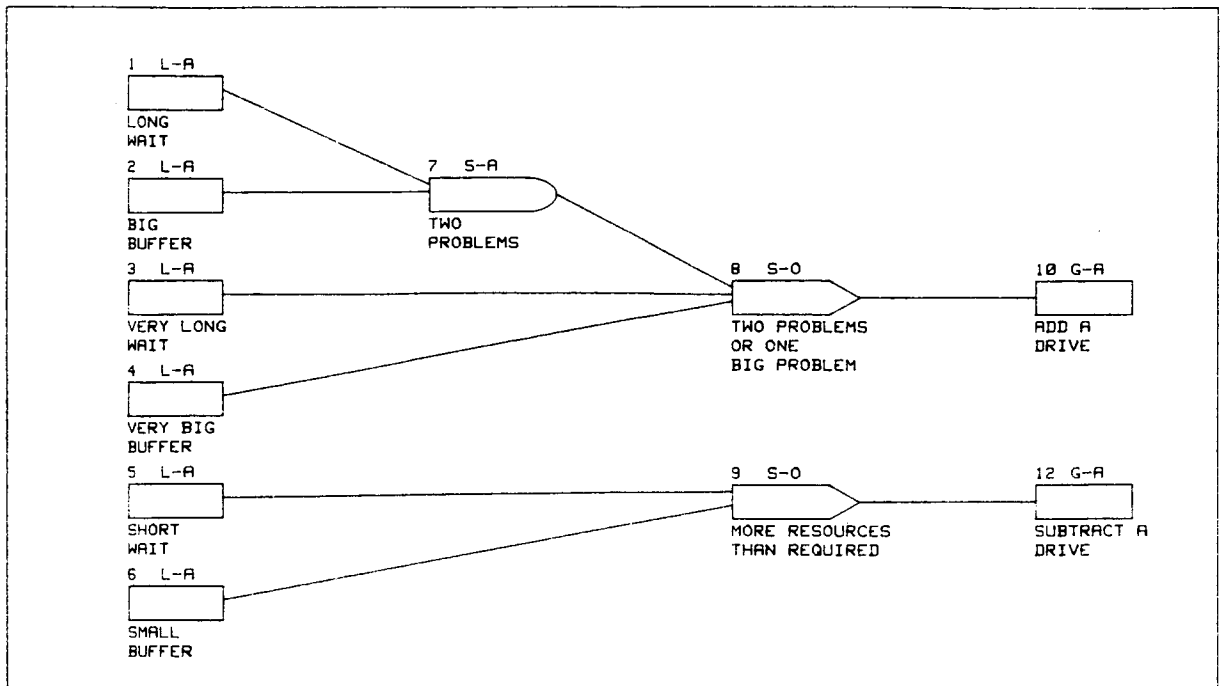


Figure 5 - Knowledge Based Used To Tune Storage Model

USING A SIMULATION TO VALIDATE AN EXPERT SYSTEM

A novel, but powerful concept, is to use a simulation to validate an expert system. An example is shown here for the case of an expert system which performs fault diagnosis. An oversimplified FAST model of a spacecraft is provided here:

```

IF Heater_On
  Inside_Temp = Inside_Temp
                + Constant * (Heater_Temp-Inside_Temp)
ELSE
  Inside_Temp = Inside_Temp
                + Constant * (Outside_Temp-Inside_Temp)
END IF
IF Thermostat_Works
  IF Heater_On AND Inside_Temp > Therm_Hi_Set
    Heater_On = 0      -- Turn Heater Off
  END IF
  IF NOT Heater_On AND Inside_Temp < Them_Lo_Set
    Heater_On = 1      -- Turn Heater On
  END IF
END_IF
  
```

ORIGINAL PAGE IS
OF POOR QUALITY

The expert system which is used to diagnose thermostat faults is illustrated in Figure 1. The values which are provided to FAIE through the interface represent the spacecraft telemetry. The job of the expert is to diagnose spacecraft faults based on the telemetry. In this case, the simulation interactive interface may be used to introduce faults on the spacecraft. For example, a simulation command could be used to "break" a thermostat. According to the model, the effect of this would be that the heater status would not change, and the temperature would not be controlled. The expert system should, in turn, recognize this fault and recommend corrective action.

CONCLUSIONS

This project has demonstrated a generic capability to operate a simulation and an expert system in parallel. FAST and FAIE are both general purpose tools. This flexibility allows for two distinct types of problem solving using the simulation and expert system combination. The first is to have the expert system fine tune the simulation model to the system specifications. The second is for the simulation to actually validate and verify the correctness of the expert system. Thus, users may build models, and related knowledge bases, and have a powerful, general purpose structure to provide the interaction.

BIBLIOGRAPHY

Ghezzi, C. and Jazayeri, M. (1982). Programming Language Concepts John Wiley and Sons, New York.

Henriksen, J.O. (1983). The integrated simulation environment (simulation software of the 1990s). Operations Research 31, 1053-1073.

LaVallee, D.B. (1986). An Ada Inference Engine for Expert Systems. Proceedings of the First International Conf on Ada Programming Language Applications for NASA's Space Station, June, 1986.

Lavery, R. Greer, Artificial Intelligence and Simulation: An Introduction. Proceedings of WSC, Dec 8-10, 1986.

O'Keefe R. (Jan.86) "Simulation and Expert Systems - A Taxonomy and some examples". Simulation, Vol46, 1, pp 10-16.

Spiegel, J.R. (1987). Interactive discrete event simulation in Ada. Proceedings of the Joint Ada Conference: Fifth National Conference on Ada Technology and Washington Ada Symposium, March 16-19. pp 121-125.

Spiegel, J.R. and LaVallee, D.B. Using an expert System to Drive a Simulation. Proceedings of the Conference on AI and Simulation, April 18-21, 1988.

Development Tools / Methodologies

**The Advice Taker/Inquirer, A System For High-Level Acquisition
Of Expert Knowledge**

**Lisp Object State Saver(LOSS): A Facility Used To Save Partial
Schedules Of The Hubble Space Telescope**

**Verification and Validation Of Rulebased Systems For Hubble
Space Telescope Ground Support**

**The Need For A Comprehensive Expert System Development
Methodology**

The Advice Taker/Inquirer, a System for High-Level Acquisition of Expert Knowledge

Robert F. Crompt
Science Applications Research, Inc.
National Space Science Data Center
Greenbelt, MD 20771

ABSTRACT

The Advice Taker/Inquirer (AT/I) is a domain-independent program that is used to construct, monitor, and improve an expert system. In the learning phase, an expert teaches a strategy to the AT/I by providing it with declarative and procedural knowledge, expressed in the expert's domain-specific vocabulary. The expert can modify any advice given to the system earlier, and any advice dependent on the altered advice is reviewed automatically for syntactic and semantic soundness. This paper discusses knowledge acquisition and methods for ensuring the integrity of the knowledge base in an expert system.

1. Introduction

This paper discusses a portion of the design of the Advice Taker/Inquirer (AT/I) [2], a domain-independent program written in Common LISP that interactively builds expert systems. The system is conceptualized in two phases: the learning phase, reported here, accepts advice from the expert in the form of declarative and procedural knowledge, can be taught some of the domain dependent vocabulary, and alerts the expert to any effects modifications on one portion of the strategy have on other parts; and the operational phase, during which the AT/I monitors and suggests improvements to the expert system's problem solving strategy based on the system's performance and analysis of an expert's criticism.

McCarthy first proposed an advice taker [4], and various researchers have formalized, expanded upon, and implemented some ideas in this area [6,8]. This version of the AT/I is the result of the author's Ph.D. research on knowledge acquisition and machine learning, and its preliminary approach is based off Findler *et al.*'s domain-specific implementation of a poker playing AT/I [3].

Section 2 discusses some of the major modules in the learning phase of the AT/I. Throughout the paper, the terms *expert* and *user* are used synonymously since the user of an expert system building tool is indeed an expert. This contrasts with the user of the resulting expert system, whose level of proficiency can conceivably be anything from novice to expert.

2.1 The Definition Module

Object definition refers to the task of teaching the Advice Taker/Inquirer about the general structure of a class of objects. By using the definition module the user constructs a template that describes an object's make-up. Often, it is convenient to abbreviate the phrase "object's make-up" with the word "object," and this convention is adopted here.

The creation of an object definition is roughly analogous to declaring a type in conventional programming languages, such as Pascal. Continuing the analogy, just as a programmer declares variables in terms of the defined types, in the AT/I the expert defines entities which are instantiations of the defined objects. The instantiation module is described in Section 2.3.

The information structure chosen for representing objects is the frame [5]. The frame for some given object is a collection of information about the attributes that describe the object. Minimally,

for each attribute the frame contains knowledge about its data type, its range of permissible values, and prototypical or default values which can be used if reasoning with partial knowledge is necessary.

Many existing expert system building tools use the frame convention [1]. Passing frame information among these systems requires transforming one system's internal representation of the frame into the receiving system's format and defining a communication protocol which facilitates the interchange of data.

Knowledge Acquisition

In the AT/I, two modules are used to define a frame completely for a given object. First, the user invokes the definition module to describe the object's attributes, their data types, range of permissible values, and vocabulary which is special to this object. Second, the user can supply prototypical values for any of the object's attributes by executing the prototype module as described in Section 2.2.

The expert creates or modifies an object during the learning phase by entering the command **DEFINE <object>**. If a new object is being defined, the Intelligent Screen Editor (ISE), a general utility developed for the AT/I, displays an empty shell. As the user supplies information, additional questions relevant to the definition of the specific object are added to the screen. If an existing object definition is being modified, then ISE presents header information about the object and a list of its attributes. In the latter case, by selecting an attribute the user causes ISE to make its definition visible and available for being changed.

An object's frame consists of one or more attribute definitions. When only one attribute is necessary to convey all the relevant information about an object, it is more convenient to let the object's name refer to the value stored under the single attribute of that object. For example, we could define the object *grocery bag* and the only attribute we may be interested in is its *contents*. Unambiguously, we can use the phrase *grocery bag* and understand its connotation to be "the *contents* of the *grocery bag*." The name of the object serves as a descriptor for some quality about the object.

When only one attribute constitutes the frame, and the user decides that the object name should reference the value stored for that attribute, then the class of the object definition is termed a descriptor; otherwise, the class is known as a record. An attribute-object pairing is also termed a descriptor.

The user selects either *record* or *descriptor* for the object definition's class. If *record* is chosen, then ISE requests the name of each attribute and its definition until the user has completed defining the object. If *descriptor* is selected, then the user is allowed to proceed directly with the definition.

The first thing required in a definition is the attribute's or descriptor's type. The AT/I aids in constructing a type definition by furnishing broad type templates which the expert then customizes with respect to the attribute or descriptor being defined. The AT/I accepts numerical, ordered categorical, unordered categorical, list, and user-defined types, and each of these is described now.

Numerical. The numerical type should be used if the value for the named attribute or descriptor is a single number or measurement. If this type is chosen, then the user must supply the lower and upper bound on the numerical range. The symbols **-inf** and **+inf** ("negative infinity" and "positive infinity," respectively) can be used if the range is open on either end or if there is no meaningful cut-off point. Thus, a finite range, a range bounded at one end, or a totally open range

can be defined. Any constraints within the range, that is subintervals or points which should not be included in the range of this definition, can also be furnished.

In addition to supplying the range of values, the user is asked to provide, if it exists, the minimum measurable difference between two points along the range. This is termed the maximum meaningful resolution (MMR), and if it is supplied then all values given for a decision variable which is of this type are assumed to be the result of applying some measuring tool which has a finite degree of precision. A user-supplied value which involves a higher degree of precision than defined would indicate that the user has either misread the measuring instrument, mistyped the value, or committed some other error.

Unless the quantity represented by this type is dimensionless, a complete type definition requires the unit of measurement used for the range and MMR. Information on converting from one unit of measurement to another can be supplied at another time by invoking the unit definition module, described in Section 2.6.

Finally, a numerical type definition is completed by specifying English expressions of comparison which are synonymous with the phrases "is less than," "is equal to," and "is greater than." These context specific phrases allow the expert to express his advice more naturally, aid in the parsing of production rules, and can be used by the expert system itself when explaining how or why a certain action occurred.

Frame name: STAR

Class: DESCRIPTOR or RECORD

...

Attribute: **DISTANCE**

Type: **NUMERICAL**, ORDERED CATEGORICAL, UNORDERED CATEGORICAL,
LIST, USER-DEFINED

-- Acceptable range of values --

Lower bound: **0**

Upper bound: **+INF**

-- Constraints on range --

Are there any constraints on this range? **NO** or YES

Maximum meaningful resolution: *****

-- Unit of measurement --

Unit of measurement (singular): **PARSEC**

What is the plural form of PARSEC? **PARSECS**

-- Expressions of comparison --

< IS CLOSER THAN, IS NEARER THAN

= IS AS CLOSE AS, IS AS FAR AWAY AS

> IS FURTHER THAN

Figure 2.1: Definition of a numerical type.

Ordered categorical. A type should be declared ordered categorical if its range of possible values consists of a finite number of literal or symbolic values and a meaningful ordering occurs among these elements. This type is defined by furnishing all the possible values in the correct order, and supplying expressions of comparison for "is less than," "is equal to," and "is greater than."

```

...
Attribute: SPECTRAL CLASS
Type:      NUMERICAL, ORDERED CATEGORICAL, UNORDERED CATEGORICAL,
          LIST, USER-DEFINED
-- Ordered values --
    Ordered values: O, B, A, F, G, K, M
-- Expressions of comparison --
    Fill in the blank with phrases which express the relation. Separate
    individual phrases with commas.
    O _____ B: IS A HOTTER SPECTRAL CLASS THAN
    O _____ O: IS IN THE SAME SPECTRAL CLASS AS
    B _____ O: IS A COOLER SPECTRAL CLASS THAN

```

Figure 2.2: Definition of an ordered categorical type.

Unordered categorical. If the range of possible values consists of symbolic values which have no meaningful ordering among themselves, then the type should be declared as unordered categorical. As in the ordered categorical values, the user must supply a list of all possible values. Expressions of comparison are requested for the phrases "is equal to" and "is unequal to."

List. If the value for a named attribute or descriptor is a set, then its definition is of type list. A list type definition is created by stating the size of the list if it is of fixed size, and the type of element within the list itself. The element type need not be defined prior to referring to it.

```

Frame name: CONSTELLATION
Class:      DESCRIPTOR or RECORD
Type:      NUMERICAL, ORDERED CATEGORICAL, UNORDERED CATEGORICAL,
          LIST, USER-DEFINED
-- Definition of list --
    Type of element in list:  NUMERICAL, ORDERED CATEGORICAL,
                              UNORDERED CATEGORICAL, or USER-DEFINED
    Name of user-defined type: STAR
    Size of list (enter * if no fixed size): *

```

Figure 2.3: Definition of a list type.

User-defined types. In addition to customizing any of the above type templates, the user can define a type by using or modifying a type which has been created previously. To do this, the name of the new type is declared, and then the name of an existing type is given. Once a frame has been defined, it becomes a valid type and can be used to define another object, an attribute which modifies an object, or a descriptor. Similarly, an existing definition of an attribute for some object can be referenced by entering <attribute> OF <frame> when asked for the existing type's name. Finally, the element type of a list type can be selected at any time by entering either MEMBER OF <descriptor> or ELEMENT OF <descriptor>, where <descriptor> is either a descriptor or an attribute of some object, and <descriptor> is of type list.

The expert can opt to copy or share the definition of the chosen type. If the definition is copied, then any references within the existing definition to itself are altered to refer to the new type in its version of the definition. If the definition is shared, then a master-slave relationship is created between the existing and new type, and any changes made to the master type are propagated to its slaves. Synonymous types can thus be created (see Figure 2.4). The slave cannot be altered independently of the master while the master-slave link is active. Any existing type, including a

slave, can serve as master to a new type. When a master type is altered, the slaves at all levels below it are affected. A slave can be freed from its bonds by changing the copy/share facet of its definition to "copy," in which case the definition of the absolute master is copied, excluding references to its slaves.

Thus, within a frame definition an attribute can be defined in terms of another frame. This embedding of frames can occur to any level, limited only by its utility and comprehensibility.

Modification of a Definition

The expert can modify an existing definition by invoking ISE on the object. If the object is a descriptor whose type template is numerical, ordered or unordered categorical, or list, then the various properties known about the descriptor are displayed and made available for immediate alteration. If the class of the object is a record, then a list of its attribute names is displayed, and by selecting an attribute, its definition is expanded upon the screen and it can then be altered. If the definition of a descriptor or selected attribute is a user-defined frame, then ISE suspends editing on the current level of the object and presents the next level definition of the selected attribute or descriptor for modification. When the editing at a lower level is exited, processing at the next higher level is resumed.

A new attribute can be added to a record during an editing session, and an existing attribute can be deleted provided it is not the master in some master-slave relationship. Currently, the only way available to delete a master type definition is to copy its definition to one of its slaves, and then transfer its remaining slaves to subservience under the copied definition. Deletion can then occur because the type no longer has any slaves dependent on its definition.

Frame name: **GALAXY**
Class: **DESCRIPTOR** or **RECORD**
...
Attribute: **DISTANCE**
Type: **NUMERICAL, ORDERED CATEGORICAL, UNORDERED CATEGORICAL, LIST, USER-DEFINED**
Name of user-defined type: **DISTANCE OF STAR**
Status of definition: **COPY** or **SHARE**

Figure 2.4: Creation of a synonymous type.

Knowledge Representation

The structure of the frame, represented as a list of descriptors, is stored on the object's property list. In turn, information about the definition of each descriptor is stored under that descriptor's property list. Since the same attribute can have different definitions for different objects, a unique descriptor name is formed by concatenating the object name, a colon, and the attribute name. Exemplary internal representations are shown in Figure 2.5.

a) Representation of a descriptor.

CONSTELLATION

text	constellation
definition_type	descriptor
frame_structure	(constellation)
frame	constellation
valtype	list
element_type	star
size	*

b) Representation of a record.

STAR

text	star
definition_type	record
frame_structure	(star:temperature star:absolute magnitude star:apparent magnitude star:distance star:position star:spectral class)

c) Representation of an attribute-object pairing.

STAR:DISTANCE

frame	star
valtype	numerical
MMR	*
range	(0 +INF)
constraints	nil
units	(parsec parsecs)
<	("is closer than" "is nearer than")
=	("is as near as" "is as far away as")
>	("is further than")
slaves	(galaxy:distance)

d) Representation of a synonymous type.

GALAXY:DISTANCE

frame	galaxy
valtype	star:distance

Figure 2.5: Internal representations for a) a descriptor; b) a record; c) an attribute-object pairing; and d) a synonymous type.

2.2 Prototypes

The prototype for an object is an ideal instantiation of the object, a composite view of all entities that are possible instantiations of the object. A prototype is an image evoked to fill the gap when specific details are missing. Prototypes allow us to reason with default values when there is incomplete information for solving a problem.

The AT/I supports the creation and use of prototypes. To create a prototype, the command **PROTOTYPE** *<prototype>* is entered. The expert is then asked the name of the object that defines its structure. There is no limit on the number of prototypes that can be created from a given object. In effect, a prototype can be viewed as a stereotypical view for some subset of the entities that can stem from an object. For example, WHITE DWARF and RED GIANT could be useful prototypes for the object STAR.

The expert defines the prototype by supplying default values for the attributes of the object. If it seems unnatural to pigeonhole an attribute with a value then none need be given. The process of interacting with the AT/I to fill in an object definition is described in detail in the next section.

2.3 Entity Instantiation

An actual entity in the environment is created by providing real values for the slots of a frame. The user invokes the entity instantiation module by issuing the command **ENTITY** *<entity name>*. If the entity already exists, then its definition is made available for modification; otherwise, the system inquires which object definition this entity is to be patterned after, and then places the user in the intelligent screen editor. The screen displays the name of the entity, the name of the frame, and places the name of each attribute for which a value must be supplied on consecutive lines. The user can supply a value for a given slot if it is known, or the slot can be left empty, which signifies that the value is not known or is to be computed by the system during the operational phase. The acceptable values for a given slot depend on the type of template used to define the slot, as follows:

- **numerical**: If the slot requires a measurement, then the default unit of measurement is displayed to the user, and is used if the user does not override it. However, the user can enter a numerical value and a unit of measurement, in which case the supplied measure is converted internally to the expected unit of measurement, if possible. Provided the two units are in the same family of measure (*e.g.*, *foot* and *mile*, but not *foot* and *pound*), the converted value is checked to ensure it falls within the acceptable range of values, and is not in a subinterval which has been excluded from this range. Finally, if the MMR has been defined for this slot, then a warning is generated if the user-supplied value contains more digits of precision than signified possible by the MMR.
- **ordered/unordered categorical**: The user must supply a value chosen from the domain of possible values for this slot.
- **list**: The user must supply a list of values of the specified element type. Each value in the list is checked for validity. If the list is of fixed size, then that number of values must be given. It is permissible to repeat values within the list, and the ordering used by the user is retained. If no value is entered for this list and a null list is legal in this context, then the user is asked whether this denotes the list is empty, or the value is unknown.
- **user-defined**: The name of an entity which is an instantiation of the slot's type must be entered. If the entity supplied has not been created yet, then the system generates a warning, and internally places the name of the supplied entity on a special structure known as the *agenda*. In general, an item is placed on the agenda if it is referred to by the user prior to being defined to the AT/I. Any information the system can deduce about the undefined item is stored as *notes*, and these notes are cross-referenced for consistency if the item is referred to elsewhere prior to its definition, and at the time of its definition. If a conflict is detected, then the user must

select the correct usage of the item, and all instantiations which incorrectly reference the item are marked erroneous and must be fixed. When a slot's value is flagged as invalid, then the system functions as if the value is unknown.

2.4 Production Rules

Structure of a Rule

A production rule serves as the basic form of procedural knowledge in the AT/I. Structurally, a rule consists of an antecedent and a consequent, and only if the antecedent is true is the rule possibly fired and the consequent subsequently executed.

Specifically, a rule has the form:

IF <clause> {AND <clause>}* THEN <statement> {AND <statement>}*
 where <clause> is a logical test involving two expressions, such as "luminosity of the star is brighter than 10 solar units" or "the pulsar is in the crab nebula," and <statement> is a directive built out of AT/I primitive actions and expressions in the user's domain that have previously been taught to the AT/I (see Figures 2.6 and 2.7).

<u>Structure</u>	<u>Example</u>
literal	red
number	5
measurement	1.523 AU
<i>descriptor</i>	constellation
<i>object</i>	star
<i>entity</i>	milky way
function	orbital eccentricity
function-arguments	orbital eccentricity of Mars
<i>attribute-object pairing</i>	position of the star
<i>attribute-entity pairing</i>	red shift of the crab nebula
attribute-function-arguments	position of the brightest star in Libra

Figure 2.6 Structures for creating expressions. Italicized structures are termed *decision variables* because they express the status of the environment and are the basis for deciding on subsequent actions.

Acceptance of a Rule

When the user is defining a production rule, each clause or statement is verified both syntactically and semantically as soon as it is furnished. If no error is found, then the system silently allows the user to move on in the rule's definition. However, if the clause or action does not parse, then a detailed error message is generated and displayed in a pop-up window, and the felonious portion of the rule is highlighted. The user can either fix the error right away, or ignore it for the moment and furnish another part of the rule. In any event, the entire rule must parse prior to the system accepting it.

The syntactic parse is first performed, and only if it succeeds is the semantic verification undertaken. A definite clause grammar [7] is used to capture the syntactic structure of the clause or statement. The non-terminal nodes in the grammar are built-in to the AT/I since the syntactical structure of the statements is fixed. In addition, new non-terminal nodes are necessary whenever the expert defines a domain-specific function or procedure because the grammar must incorporate the function's or procedure's arity. The grammar is enlarged automatically as the AT/I acquires new knowledge.

- *<Grammar for primitive action>*
 - *<Example of usage of primitive>*
- **<variable> = CREATE ENTITY <object, entity, or prototype>**
 - NEW CHART = CREATE ENTITY STAR CHART
- **<variable> = {THE} [LAST | FIRST] <element> [OF | IN] <list>**
 - MOST DISTANT OBJECT = THE LAST OBJECT IN SORTED RED SHIFTS
- **<variable> = <elements> OF <list> THAT SATISFY <conditions>**
 - RED GIANTS = STARS OF STAR PLATE 23 THAT SATISFY (LUMINOSITY OF STAR IS GREATER THAN 1000 SOLAR UNITS, TEMPERATURE OF STAR IS COOLER THAN 6000 K)
- **<variable> = PARTITION <list> ACCORDING TO <constraints>**
 - STAR SETS = PARTITION STAR PLATE 23 ACCORDING TO SPECTRAL CLASS
- **FOR EACH <element> [OF | IN] <list> PERFORM <actions>**
 - FOR EACH STAR IN THE BIG DIPPER PERFORM (PUT THE STAR IN THE NEW CATALOG, REMOVE THE STAR FROM THE OLD CATALOG)
- **IF <conditions> THEN <actions> ELSE <actions>**
 - IF THE TEMPERATURE OF THE STAR IS HOTTER THAN 6000 K THEN (HOTTER = HOTTER + 1, OUTPUT STAR " IS HOTTER THAN THE SUN.") ELSE OUTPUT STAR "IS COOLER THAN THE SUN."
- **SORT <list> ON [<attribute> [INCREASING | DECREASING | + | -]]⁺**
 - SORT X-RAY SOURCES ON PERIODICITY INCREASING
- **PUT <element> IN <list>**
 - PUT BETELGEUSE IN ORION
- **REMOVE {THE} [LAST | FIRST] <element> FROM <list>**
 - REMOVE THE FIRST SET FROM STAR SETS
- **REMOVE <element> FROM <list>**
 - REMOVE BETELGEUSE FROM RED GIANTS
- **EMPTY <list>**
 - EMPTY THE CONSTELLATION
- **<procedure name> <arguments>**
 - COMPUTE THE ORBIT OF MARS
- **OUTPUT [<text string> | <evaluatable expression>]⁺**
 - OUTPUT "Perihelion of " PLANET " = " PERIHELION OF PLANET "."

Figure 2.7 Built-in primitives for constructing actions with the AT/I, and some examples drawn from astronomy.

Terminal nodes are drawn from the set of user-defined object, entity, procedure, and function names and their definitions, augmented by a small list of built-in functions and phrases of comparison.

The grammar is written so that any input will generate a parse. Further syntactic analysis of the parse either confirms the validity of the input's structure or produces a list of errors therein. If errors are found, then customized error messages are generated and displayed to the expert, and the invalid clause or statement is highlighted and internally flagged as requiring fixing.

Only after the syntactic parse is verified does the system perform a semantic check on the input. The parse tree is examined to ensure that arguments to functions and procedures are the correct type, phrases of comparison are used in the proper context, and units of measurement are consistent, for example. If the semantic parse is also valid, then the clause or statement is accepted; otherwise, suitable error messages are generated, and the input is highlighted and flagged as before.

Compilation of a Rule

When the expert has finished defining the rule, and all input has been validated, the system encodes the rule, transforming it from pseudo-English to Common LISP. Though this mapping is relatively quick (a matter of a few seconds), it is by no means trivial. References to objects are translated to LISP function variables, working variables are created for all local variables which occur in the production rule, special encoding takes place to handle recursive function calls, and decision variables are represented in a canonical form so that the inference engine can evaluate or bind them as necessary.

A decision variable (DV) in the rule can be defined with respect to an object or an entity. Any DV in the rule which is in terms of an object (i.e., there is no explicit mention of an entity) requires binding prior to the invocation of the rule at run time. When the rule is compiled into LISP, the object based DV's are translated into LISP code variables and placed in the resulting LISP function's parameter list. This mapping is stored on the rule's property list and the inference engine consults it prior to invoking the rule during the execution phase.

A list of all the decision variables found in the antecedent is also stored as a property of the rule. The *read set* of the rule is a combination of the DV's in the antecedent and the independent DV's found in the rule's consequent.

A list of the decision variables whose values are potentially altered if the rule is fired, denoted the *write set* of the rule, is also placed on the rule's property list. In a similar fashion, the property list of each referenced decision variable is updated to indicate that this rule can possibly alter its value.

Knowing the read and write sets for each rule, and which rules are activated given a change in the environment, allows the implementation of an inference engine that has the potential for performing forward and backward reasoning. Additional knowledge on the complexity of computing a value for a decision variable is necessary if an efficient reasoning mechanism is to be created.

Modification of a Production Rule

Modification of an existing production rule is initiated by giving the command **RULE <rule name>**. The rule is displayed in ISE, and any portion of its definition can be altered, deleted, or

expanded. The same syntactic and semantic verification occurs as previously described, and once the rule is accepted, the new encoding of the rule replaces the existing definition. The read and write sets are recomputed, and the rule is removed from the property list of any decision variables which occurred in the prior rendition of the rule but that are no longer in the modified version.

The rule can also be deleted from the rule set, in which case all references to it are removed from the knowledge base.

2.5 Procedures and Functions

The set of actions recognizable by the AT/I is expanded whenever the user creates new procedures or functions. The distinction between the former and the latter is that a procedure call does not return a value and so can only be used as a complete statement in a rule's consequent or within the body of a procedure or function, whereas a function call returns a value and must be used as an expression in an antecedent or consequent.

2.5.1 User-defined Procedures

Definition of a Procedure

The command **PROCEDURE** *<procedure name>* initiates the definition of a procedure. The first step in procedure definition is the supplying of the procedure's parameter names and their types. Each parameter must be specified, and the ordering in which the parameters are given defines the order in which the parameters must be supplied in any statement invoking the procedure. A given procedure must require a fixed number of parameters, and it is valid to define a procedure whose arity is zero. The parameter type must be expressed in terms of an existing type, and it is permissible to use any phrase that represents a decision variable: object, descriptor, attribute-object pairing, *entity*, or *attribute-entity pairing*. The base type of the decision variable is used as the type of the parameter. The ability to define a parameter's type with respect to an entity is not conventional computer science practice, but it is a convenience offered to the user and lends itself to a quite natural interpretation.

Procedure calls follow a call-by-name convention, so any assignment made to a decision variable defined in terms of a parameter causes the change to be made to the decision variable that it is bound to at the next higher processing level.

The body of a procedure is defined the same way as the consequent to a rule is defined. The decision variables must be defined in terms of known entities or the procedure parameters--the procedure parameters are treated as the only valid objects recognizable within the procedure body. Recursion is fully supported, so a procedure can call itself either directly or indirectly.

Encoding of a Procedure

The procedure name is added to a list of user-defined procedures and functions, its arity is recorded, and a tag is attached indicating that this is a procedure. The parameters and their required types are stored on the property list of the procedure, and mapped into LISP variables for the actual LISP function definition (everything in LISP is a function, according to the language definition--the value returned by the LISP function is simply ignored by the AT/I in this case since this encoding represents a user-defined procedure). The body of a procedure is encoded the same way that a consequent for a rule is encoded.

The write set for the procedure is determined also and placed on the property list. The list of entities and objects which are potentially activated given the execution of the procedure must be known so that the write set of the higher level process which invokes this procedure can be ascertained correctly.

Modification of a Procedure

If the definition of a user-defined procedure is altered, then the procedure's old write set is replaced by the new write set. If the arity of the procedure is changed, or if the type of a parameter is modified, then any rules, procedures, or functions which contain calls on this procedure must be updated to reflect this change. The agenda mechanism tracks the required alterations. The AT/I flags any affected procedural knowledge, and alerts the user of the required fixes. Any flagged definitions are made inactive until the user remedies the no longer proper procedure call and the item is removed from the agenda.

The user can also delete the entire procedure, in which case the procedure name is removed from the list of user-defined procedures and functions, and the agenda mechanism records those production rules, procedures and functions whose definitions reference the purged function. Alternately, the user can supply a new procedure definition for this procedure name, and if its parameter list has the same arity and ordering of parameter types, then the flag placed on procedural knowledge due to the deletion of the procedure is removed. The agenda mechanism also oversees this task.

2.5.2 User-defined Functions

Definition of a Function

The command **FUNCTION** *<function name>* is used to create or modify a user-defined function. As in a procedure definition, the user must specify the parameters to the function and their types, but in addition, the function return type must also be given. In the function body, only entities, the function parameters, and the function name can be used as the basis for decision variables. The function name must be assigned a value some time during the body of the function because the final value assigned to the function name is the value returned by the function, and it is this value which is subsequently used in the next higher processing level.

A problem arises when a call-by-name convention is adopted with respect to function calls. With a production rule control structure, the inference engine can test whether the left hand side of a rule is satisfied prior to executing its right hand side. In testing the truthhood of the left hand side, a clause could be encountered which contains an expression involving a user-defined function call. Let us assume that this function call involves some parameters, and that as a result of the function executing, one of the parameters has its value changed. If a call-by-name protocol is followed, the decision variable within the rule that is bound to the function parameter is altered as a consequence. But, if one of the clauses of the antecedent is subsequently discovered to be false, then the inference engine concludes that the rule should not be fired, and so it should have no effect on the environment at this time. This problem of erroneously altering the environment can be circumvented by knowing prior to the evaluation of a rule the set of decision variables whose values might be modified as a direct result of the rule's execution. Fortunately, this information is readily available since it is stored as the write set on the rule's property list. Thus, immediately preceding the testing of the rule's antecedent, the values for the decision variables in the write set are saved. The antecedent is then tested, and if it fails, the decision variables in the write set are reset to whatever values they had at the start of the rule's examination, eradicating any of the rule's unwarranted side-effects.

Encoding of a Function

The same method for encoding a procedure is followed in creating the LISP version of a user-defined function, but in addition, the LISP function is coded so that its return value is the value assigned to the user-defined function name. The list of user-defined functions and procedures is updated to include this function, the number of parameters and their types are saved on the function name's property list, the write set for the function is noted, and the definition is classified as a user-defined function.

Modification of a Function

The routine as outlined above for procedure modification is followed if any portion of a user-defined function is altered, including the function return type.

2.6 Units of Measurement

Every domain which is complex enough to warrant the creation of an expert system for solving some of its problems undoubtedly requires the expert at some point to give advice on reasoning about some numerical data within the domain. Unless the data are purely subjective, as in some psychology experiments, the values must be qualified with respect to some unit of measurement. Generally, a unit of measurement belongs to a family of measurements, where different units in the same family measure along the same dimension but employ different scaling factors, such as *foot* and *mile*. The unit of measurement is just as important as the numerical value in most cases (*e.g.*, the unit of measurement chosen in a given dimension has no effect on the numerical value 0, unless the dimension is *temperature* which is weird anyway because conversion from one unit to another is not strictly multiplicative, but also involves the addition of a constant offset).

As previously discussed, the expert supplies the unit of measurement when a numerical type is defined. Related units of measurement in the same metric family are taught to the AT/I by using the unit definition module. This module is also used for acquiring the definition of more complex units, referred to as *derived units*, which involve a combination of metric families (*e.g.*, *mph* = *mile / hour*).

The command **UNIT** invokes the unit definition module. The user can then define a unit in the same family as another unit by entering a statement which gives the relative scaling factor between the two; or a derived unit of measurement can be defined by entering the equivalent formula for which the unit is an abbreviation. The AT/I requests the singular and plural forms of all units referenced in the unit definition module.

- a) unit definition> **3 feet = 1 yard.**
 - Is it true that 1 YARD equals 3 FEET? **yes**
 - What is the singular form of FEET? **foot**
 - What is the plural form of YARD? **yards**
- b) unit definition> **newton = kilogram * meter / (second * second)**
 - Is this conversion formula correct:
NEWTON = (METER * KILOGRAM) / (SECOND * SECOND) ? yes
 - What is the plural form of NEWTON? **newtons**
 - Is SECONDS the plural form of SECOND in this case? **yes**
 - What is the plural form of METER? **meters**
 - What is the plural form of KILOGRAM? **kilograms**

Figure 2.8 a) Definition of two units in the same metric family.
b) Definition of a derived unit of measurement.

Encoding of Unit Definitions

The singular and plural forms of each unit of measurement are kept on a list saved in the knowledge base. Under each unit, a list of the conversion factors for switching to other units of measurement in the same family is maintained. When a new unit is learned in a particular family, the conversion factors for each unit in the family are computed, and each unit's table of conversions is augmented to incorporate the new unit. Thus, if the relation between *foot* and *yard* is first taught, and this is followed by a statement on the relation between *yard* and *mile*, the AT/I immediately computes the scaling factor that links *foot* to *mile*.

Depending on how units are defined to the system, it is possible for the AT/I to not realize that what it is representing as two (or more) distinct families of measurement are really instances of subfamilies of the same metric family. Only if the expert defines the conversion from a unit in the one subfamily to a unit in the other subfamily can the AT/I consolidate the two previously distinct groupings into one.

To aid the expert, if the command **FAMILIES** is entered either at the AT/I command level or inside the unit definition module, the system generates a list of what it considers unrelated families of measurement. If the expert notices that the system has separated the units in a family, then a definition should be supplied to unite the subsets.

If a unit is an abbreviation for some combination of units, then the formula that expresses this definition is stored on the unit's property list. If more than one formula is given for a unit of measurement, then only the most recently supplied formula is retained. There is no reason to save the previous formulae because each is equivalent in meaning. (The system could use this knowledge as a method to detect subfamilies, perhaps, but this has not been implemented.)

Formulae are represented in a canonical form, based on the observation that a derived unit when represented in terms of other units can be expressed in the form of a fraction where the numerator and denominator are comprised solely of products of units. In a formula, the numerator or denominator might also have the value 1, as shown in Figure 2.9.

<u>Derived Unit</u>	<u>Canonical Form</u>
joule	(newton * meter) / 1
newton	(kilogram * meter) / (second * second)
hertz	1 / second

Figure 2.9 Representation of the definitions of derived units in canonical form.

A nice property of this canonical form is that it is closed under multiplication and division of units, so any derived unit expressed in terms of other units will result in a formula in canonical form. This property permits elegant coding for computing conversion factors from any unit of measurement to another (including between different families of measurement). Thus, as one example, the AT/I can find that to convert from *feet per second* to *miles per hour*, a value should be multiplied by 15/22.

How the AT/I Uses Units of Measurement

There are two areas during knowledge acquisition where knowledge of units of measurement is of pivotal importance. The first place is in the entity instantiation module when a numerical measurement is required. The system displays the default unit of measurement defined for the type of decision variable being instantiated, and if the user supplies a value without qualifying it with a unit of measurement, the system assumes it is in terms of the default unit of measurement. However, if the user enters a value and a unit, then the system first checks that the user's metric is in the required family, and if it is, then the value is converted into the default unit of measurement automatically and tested against the valid range for the decision variable. The user is informed if the unit is not recognized or not known to be in the same family as the default unit of measurement.

The second area where units of measurement are employed is during the parse stage of the production rule, procedure and function definition modules. If a clause in an antecedent involves units of measurement, then the two expressions connected by some relational operator must both evaluate to values which are in the same family of measurement. Similarly, if a decision variable is assigned some value as the result of executing some statement that contains metric units, then both the DV and the expression that defines its assigned value must be in terms of measurements from the same family. The expert's advice is not accepted if it does not pass this uniformity requirement.

ACKNOWLEDGEMENTS

The author would like to thank Nick Findler and the members of the Group for Computer Studies of Strategies at Arizona State University for their comments and encouragement throughout the years. A word of thanks to Bill Campbell, Nick Short, Scott Wattawa and Larry Roelofs, members of the NSSDC Applied Artificial Intelligence Laboratory at NASA/GSFC, for initiating me into their ways and giving the AT/I a home.

REFERENCES

- [1] Becker, S. and Selman, B., An overview of knowledge acquisition methods for expert systems, University of Toronto Computer Systems Research Institute Tech. Rep. CSRI-184, 1986.
- [2] Crompt, R.F., The task, design and approach of the advice taker/inquirer system, Arizona State University Department of Computer Science Tech. Rep. TR-85-014, 1985.
- [3] Findler, N.V., Sicherman, G. and Feuerstein, S., Teaching strategies to an advice taker/inquirer system, in: P.A. Samet (Ed.), *EURO IFIP 79*, North-Holland (1979) 457-465.
- [4] McCarthy, J., The advice taker, in: M. Minsky (Ed.), *Semantic Information Processing*, Cambridge: MIT Press (1968) 403-410.
- [5] Minsky, M., A framework for representing knowledge, in: P. Winston (Ed.), *The psychology of computer vision*, New York: Mc-Graw-Hill (1975) 211-277.
- [6] Mostow, D. J., Mechanical transformation of task heuristics into operational procedures, Ph.D. dissertation, Carnegie-Mellon University, 1981.
- [7] Pereira, F.C.N. and Warren, D.H.D., Definite clause grammars for language analysis--a survey of the formalism and a comparison with augmented transition networks, *Artificial Intelligence* 13 (1980) 231-278.
- [8] Zobrist, A.L. and Carlson, Jr., F.R., An advice-taking chess computer, *Scientific American* 228 6 (1973) 92-105.

Lisp Object State Saver (LOSS)
A Facility Used to Save Partial Schedules
of the Hubble Space Telescope

Jeffrey L. Sponsler
Space Telescope Science Institute ¹
3700 San Martin Dr.
Baltimore, MD 21218

Abstract

Current research in the area of long term scheduling of the Hubble Space Telescope is being done using Common Lisp and Flavors on Lisp Machines. The planning tools manipulate memory-resident data structures which represent the many entities and relationships that represent planning states. The Lisp Object State Saver (LOSS), a general purpose utility, has been constructed which allows one to take a *snapshot* of memory by storing a representation of the structures in a text file. This text file can later be loaded thus restoring the pre-existing and logically equivalent planning state. A LOSS template must be created for each datatype to be stored and a simple grammar governs the creation of such templates.

¹Operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration

PRECEDING PAGE BLANK NOT FILMED

1 Introduction

1.1 What is it?

The Lisp Object State Saver program was designed as a mechanism that allows one to store memory-resident data structures in a LISP environment to an ASCII file. This file can later be loaded into memory which would be restored to a logically equivalent pre-save state.

At first glance this is not much of a problem for, say, a simple variable x that is bound to an integer. However, certain data structures often have references to other more complex data structures (eg., LISP *structures* and *flavors*). Such a reference is a pointer (internal address) and therefore cannot easily be stored as character data that would be meaningful upon memory restoration.

1.2 Why would one want to use such a facility?

In a LISP environment, it is often the case that one creates large numbers of data structures that are organized into networks with rich and complex interconnections. Such networks have been used to represent knowledge about a wide variety of domains (e.g., medicine, factory scheduling, etc.). The data structures themselves represent the nodes (entities) and their attributes store pointers to other data structures and thus represent the arcs (relationships) in a directed graph model. Such memory resident knowledge bases are the rule in artificial intelligence applications. The creation of semantic networks generally is accomplished by loading a file that contains LISP forms that upon evaluation instantiate the data structures. Programs can then be executed that (as side effects) will create logical interconnections that dynamically form the network(s). Often this can take a considerable amount of real time. In many LISP environments, once such a large and richly connected memory state is formed, it is advantageous to save the entire program memory state to a binary file on a hard disk medium (as opposed to recreating all structures and connections again). This file later can be reinstated as the memory of the LISP interpreter.² Although this works well there are disadvantages.

1. If one does not effect garbage collection before the save, a large amount of disk space will be used to save uncollected garbage (memory locations that are no longer referenced).
2. These binary files are often huge and it is not possible to store very many of them.
3. It may be impossible to take the binary image file from one lisp environment (e.g., an Explorer) and load it into another (a SUN).
4. This mechanism stores not only the relevant knowledge base but also the Lisp interpreter, and data structures that are associated with the operating environment but irrelevant to the precise application that is the focus of one's work.

²On a LISP machine, the file is a bootable band; on a conventional architecture, the file is an executable image.

With the above problems in mind, this system has been designed and implemented.

1.3 LOSS is a tool used to enhance SPIKE

The SPIKE system is being developed at the Space Telescope Science Institute to support the creation of long term schedules for the Hubble Space Telescope [1].

Briefly, the SPIKE system prototype currently resides on a Texas Instruments Explorer computer and is implemented in Common Lisp and MIT Flavors. A mouse/menu/graphics-oriented interface is provided for the user.

The central concept of the planner is the *suitability* which is implemented as a *piecewise constant function*; such a structure is a list of time points and values that represent how well an activity would schedule over time. Entities represented as flavors in the system include targets (stars, planets), activities (exposures, acquisitions, calibrations), segments of time, and constraints (sequential offsets, separation, sun exclusion, etc.). A group of activities may be grouped into a *scheduling cluster*, which is then considered a unit (the suitabilities of all activities within contribute to the cluster suitability).

By various means (manual or automatic selection) one may *commit* a cluster to a time segment. The act of commitment may often change the suitabilities of other scheduling clusters.

After a planner (a human who is working on an HST schedule) has worked with the SPIKE system for some period of time and possesses a partial schedule of some merit, the LOSS system can be invoked to store a *snapshot* of memory to a file. The planner may then continue working with the schedule that remains in memory (and may file other states). If at some point, a scheduling session has reached a point that is undesirable, there are two options. The first entails using the SPIKE planner to undo commitments in an order which is the reverse of the order in which they were made. The second option can be invoked if the planner wishes to return to a much earlier state; SPIKE operating memory is first cleared and then the LOSS State Manager is employed to select a state that had earlier been saved to file and to restore that state to memory.

1.4 How is this state saver used?

The state saver is able to save most types of data structures as logical character representations in a specified file. Currently it is possible to save the following: global variables, atoms, numbers, lists, strings, hash tables, structures, and flavor instances. The mechanisms to save arrays, property lists, and association lists are not yet in place.

It also has the ability to restore those data structures to a state that is logically equivalent (based on pointer interconnections) to the pre-save state. Only the data structures and attributes of such structures that the user specifies are saved. The file character representation that is the result of a save is generic and machine-independent; this system does however operate currently only in a Common Lisp environment with a Flavors system resident.

In order to facilitate the organization, filing, and restoration of states, a State Manager has been designed and implemented using Lisp, Flavors, and Common Windows ³.

These steps involved in using LOSS are detailed with examples in the following sections.

2 Using LOSS to save a Lisp State

Following is a brief discussion of the main elements of the operating LOSS environment. A description of how the state saver works and a simple example are included.

2.1 The State is the focus of processing.

In order to effectively manage the saving and restoring of lisp memory states, the concept of the *state* has been implemented as a data structure. A *state tree* can be created that organizes the states into a strict hierarchy. The attributes that can be associated with each state node include the following:

1. **Parent** and **child** links for tree traversal.
2. A **comment** that serves to describe the state.
3. A **state file** string that names the disk file where the state resides or will reside.
4. A **template file** where the appropriate LOSS templates are stored.
5. A **data-list** of structures to save.

2.2 The State Manager is the interface to LOSS

The State Manager is an interactive user interface that has been developed to facilitate the processing of states. It employs menus and windows and allows one to traverse the state tree, to select state nodes for processing, to create and delete nodes, and to initiate a state save or a state restore.

2.3 How does LOSS work?

To illustrate the use of this system, consider a simple example database that describes tropical fish in an aquarium. Following is the code that would define the data structures:

³Common Windows is a package that is available from Intellicorp.

```
;; Aquarium example data definitions:
(defvar *AQ* nil)
(defstruct fish name genus species aquarium)
(defstruct aquarium
  (volume 0)
  (fish-list nil)
  (shape nil))
(defmacro create-fish (name genus species)
  '(setf (aquarium-fish-list *AQ*)
    (cons (make-fish :name ',name
                    :genus ',genus
                    :species ',species)
          (aquarium-fish-list *AQ*))))
```

The following code will create an aquarium data structure and some fish structures.

```
;; Instantiate data objects into memory -
(setq *AQ* (make-aquarium :volume 30 :shape 'rectangle))
(create-fish leopard-cat corydorus julii)
(create-fish clown-loach botia macracanthus)
(create-fish neon-tetra hyphessobrycon innesi)
```

When this code has been executed, the memory resident data structures will include the following:

```
#<aquarium 41340377> is a aquarium
  VOLUME:      30
  FISH-LIST:   (#<fish 41340432> #<fish 41340421> #<fish 41340410>)
  SHAPE:      rectangle

#<fish 41340432> is a fish
  NAME:      neon-tetra
  GENUS:     hyphessobrycon
  SPECIES:   innesi
```

The data structures for #<fish 41340421> and #<fish 41340410> will also exist. (The pound-sign notation seen in these structures is a lisp convention that represents a non-printable pointer reference.)

2.4 Templates are the key to state save/restore actions.

LOSS depends upon *templates* in order to correctly guide the save/restore processes. A template is itself a data structure that contains precise information about data structures to be saved or restored. For advanced datatypes defined as structures or flavors each slot⁴ that contains a pointer to a *complex* data structure is described as well. Simpler data structures such as strings, atoms, and lists do not require explicit save/restore specifications. At the present, templates must be created manually by a software engineer who is familiar with the logical interconnections of the application data structures. The lisp forms that create the templates needed to save the aquarium database follow.

```
(create-template aquarium
  :datatype 'structure
  :no-save-slots '(shape)
  :substitution-slots '((fish-list (list-of (structure fish)))))

(create-template fish
  :datatype 'structure
  :substitution-slots '((aquarium (structure aquarium)))))

(setq *IMPORTANT-SPECIAL-VARIABLES* '((*AQ* (structure aquarium))))
```

The templates created above are used to save and restore instances of the structures *aquarium* and *fish*. Before such an instance is saved, this template is used to make appropriate decisions about what to save and how. During a restore, the template is used to guide the recreation of the pre-save data structure. Generally, templates are used to specify the datatypes of values that can be placed in slots.

The general syntax of template creation forms is:

```
(create-template <datatype name> <keys-inits>)
```

The **<datatype name>** must be the identical symbolic name of the some structure or flavor defined in the application.

The **<keys-inits>** portion of the form may contain various specifications for the template concerning the datatype. The **:no-save-slots** specification must be a list of symbolic slot names whose values the user does not want to be saved to file. In our example, the slot **shape** is not to be saved. The **:substitution-slots** specification should be a list of sublists where each sublist has the form (**<slot>** **<substitution spec>**). The **<substitution spec>** adheres to the following grammar:

⁴The term *slot* will be used to represent the attributes of structures and flavor instance variables.

```

<substitution-spec> ::=
  <structure-spec> | <flavor-spec> | (list-of <substitution-spec>) |
  <hash-table-spec> | (dotted-pair <substitution-spec> <substitution-spec>) |
  <substitution-function-spec>

<structure-spec> ::= structure | (structure <struct-name>)
<flavor-spec> ::= flavor | (flavor <flavor name>)

<hash-table-spec> ::= hash-table | (hash-table [<key-spec>] [<val-spec>])
<key-spec> ::= (key <substitution-spec>)
<val-spec> ::= (value <substitution-spec>)

<substitution-function-spec> ::=
  (substitution-fn [(save 'sav-fn)] [(restore 'rest-fn)])

```

It may be the case that a user may wish to process the contents of a slot before a save (or after a restore). LOSS templates provide for this in the form of the **substitution-fn**. The substitution function is defined by the template-designer and should be specific to a problem slot. This function is called with the contents of the slot and returns the precise lisp form that the template designer wishes to have stored. Such functions should be the exception because the substitution grammar is general and should be sufficient for most data structures.

It is possible to create specifications that guide the processing of global variables during the save and restore activities. In our example the global ***AQ*** is noted as being bound in the application environment to a structure of type aquarium.

2.5 Saving a state

To save a memory state to file, using the State Manager interface, one selects a desired state node that references the appropriate template file and state file. Selecting the appropriate menu item will initiate the saving. The State Manager will then, using the templates, store the data structures to file.

The selected state node should contain a list of pointers to data structures that should be saved. Each of these is processed in turn. Consider the variable ***AQ*** that points to the aquarium instance. The pointer cannot be saved to file so it is paired with a unique symbolic index that takes the form **index25**. Both the pointer and its index are placed into a hash-table with the pointer as key. In this way, if any other data structure references that pointer, a substitution can be made such that **index25** replaces the pointer. Since all indices are simple identifiers, one can store them as character data in a file.

The type of the structure is found to be **aquarium** and it is determined that there is a template for this structure. The aquarium structure has slots **volume**, **fish-list**, and **shape**; the

contents of these slots must be considered. The aquarium template is accessed and it is found that the slot **shape** should not be saved.

Each slot that has a substitution specification in the template is processed using that specification. Generally, pointer reference components of the contents of such a slot are replaced with symbolic indices. The contents of a slot that does not have a template substitution specification are saved to file unchanged. In our example, the **fish-list** slot has a specification that requires substitution of the slot contents. Each **fish** instance found there is replaced by a unique index.

2.6 Form of a State File

The lisp forms in a state file are generic, machine-independent, and reflect the logical links between the data items using unique index symbols. In the aquarium example, the state file would contain these forms:

```
(create-datum index25 structure AQUARIUM
  :volume 30
  :fish-list '(index26 index27 index28))
(create-datum index28 structure FISH
  :name 'leopard-cat
  :genus 'corydorus
  :species 'julii)
(create-datum index27 structure FISH
  :name 'clown-loach
  :genus 'botia
  :species 'macracanthus)
(create-datum index26 structure FISH
  :name 'neon-tetra
  :genus 'hyphessobrycon
  :species 'innesi]
(setq *AQ* 'index25)
```

3 Restoring a state file to memory

To restore data structures from a state file to memory, one uses the State Manager to select the desired state node. This node should be associated with the desired state file and appropriate template file. A menu selection will initiate the action.

The state file is loaded and data structures are created. As the data structures are recreated from the state file a new hash-table is built; for each structure, the index will be assigned to the table as a key and the pointer to the structure will be assigned as the corresponding value.

Next, the templates are consulted to determine which slots are to be processed. Any index that is found that is part of the slot's contents will be replaced with the actual pointer (that is obtained from the hash table). The result should be a restored memory state.

4 Intended improvements

Currently the major weakness of the LOSS system is that the templates must be generated manually. Future versions will include automatic or semi-automatic template generation. Arrays, property lists, and association lists will be represented as saveable datatypes.

The developers also intend to port the LOSS prototype to the Common Lisp Object System when that standard has been set. Since CLOS supports explicit typing of slots this should facilitate the automatic generation of templates. The port to CLOS will also make LOSS more universally available.

5 Acknowledgements

The author wishes to thank the following persons for their comments and ideas related to the work described in this paper: Mark Johnston, Glenn Miller, and Shon Vick.

References

- [1] Miller, G., Johnston, M., Vick, S., Sponsler, J., and Lindenmayer, K. 1988, Knowledge Based Tools for Hubble Space Telescope Planning and Scheduling: Constraints and Strategies, *Proceedings of the 1988 Goddard Conference on Space Applications of Artificial Intelligence*.

Verification and Validation of Rulebased Systems for Hubble Space Telescope Ground Support

Shon Vick
Space Telescope Science Institute¹
3700 San Martin Dr.
Baltimore, MD 21218

Kelly Lindenmayer²
Astronomy Programs, Computer Sciences Corporation

As rulebase systems become more widely used in operational environments, we must begin to focus on the problems and concerns of maintaining expert systems. In the conventional software model, the verification and validation of a system have two separate and distinct meanings. To validate a system means to demonstrate that the system does what is advertised. The verification process refers to investigating the actual code to identify inconsistencies and redundancies within the logic path. In current literature regarding maintaining rulebased systems, little distinction is made between these two terms. In fact, often the two terms are used interchangeably. In this paper we discuss verification and validation of rulebased systems as separate but equally important aspects of the maintenance phase. We also describe some of the tools and methods that we have developed at the Space Telescope Science Institute to aid in the maintenance of our rulebased systems.

¹ Operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration

² Staff member of the Space Telescope Science Institute

I. INTRODUCTION

Transformation is a rulebased system written in OPS5 which converts an astronomer oriented description of a scientific proposal into the necessary parameters needed by the planning and scheduling portion of the Hubble Space Telescope ground support system. Transformation has been in an operational phase since 1985, and responsibility for the maintenance of the system has changed more than once during this time period. This is not an uncommon phenomenon as most conventional software systems outlast their maintenance programmers, but because Transformation is a rulebased system, the traditional solutions to maintenance problems are not always applicable.

Conventional software systems take an algorithmic approach to problem solving and typically there is an explicit ordering to its functional pieces. In a rulebased system there is no explicit order to the application of rules and the order is not determined until run time and may differ with each set of data. Because the interaction of the rules is determined by the data, isolating a particular problem becomes a more difficult task - especially for the maintenance programmer who may be unfamiliar with the initial design of the system. In a previous paper a method is offered for partitioning the rulebase based on a notion of rule coupling.[8] All rules which are directly affected by a particular rule would constitute a single group. This would divide the rulebase into several smaller groups. The general approach is to identify the set of rules which are directly affected by any given rule. With this information we can construct a directed graph where the nodes represent the rules and the edges represent a direct effect. We can then traverse this graph to construct solutions to various verification and validation problems.

In this paper we will discuss the verification and validation of expert systems and how this phase of development compares to that of conventional software models. In addition, we will present and discuss some of the tools that we have developed at the Institute to aid in the maintenance of our rulebased systems.

II. VERIFICATION AND VALIDATION OF RULEBASED SYSTEMS

The common adage used when trying to illustrate the difference between verification and validation is:

Verification: "Are we building the product right?"

Validation: "Are we building the right product?"

One brute force method of validation is exhaustive testing. Control flow diagrams and other static analysis tools are used often in the verification process. But no matter what the method is, the purpose of the validation and verification stage is to assess the quality of the product. Validation by itself does not guarantee high quality and testing source code does not ensure the absence of errors (especially for rulebased systems - since the application is data driven it would be impossible to construct enough test cases to exhaustively test). Even if that were possible, there are other factors involved in producing a quality software system: efficiency, completeness, reliability and usability only begin the list.[5]

The verification process is also an important part of producing a software system of high quality. In conventional systems, static analysis may be used to determine such things as coding errors which may have not been detected by testing, poor programming practices, and departures from coding standards.

Since the design of rulebased systems does not conform to that of conventional systems, many of the conventional software engineering models also fail to apply. In the case of the validation process, how do we develop a test suite that will convincingly show that the system produces correct answers. And how does one verify an expert system - conventional static analysis methods such as control flow diagrams are not particularly useful, since control is not determined until runtime.

It is evident that conventional verification and validation tools are inadequate or inappropriate, and if expert systems are to be used with any confidence in operational environments, a set of tools must be developed which are customized to the needs of expert systems. This notion is not a radical one, and researchers in the field have made progress in this area:

Expert systems are typically validated by running several test cases on the system and calculating the percentage of correct responses. This method is not particularly accurate since it depends on which test cases were selected as well as the number of cases chosen. Work has been done in this area to maximize correctness. O'Keefe discusses validating expert systems by stratifying the test data and implementing other means of keeping test data unbiased.[10] Cragun offers a solution to the validation problem by using decision table based processing for checking completeness.[4]

Other work has been done to try and reduce the scope of the problem by partitioning a rule set based on "relatedness" - where rules are weighted and grouped according to how they interact with each other.[6]

Nguyen *et al* developed a set of verification and validation tools or LES, a generic rulebased system building tool similar to EMYCIN.[9] These tools check for problems such as conflicting rules, subsumed rules, circular rules, dead end if conditions, dead end goal rules, and other consistency and completeness properties. The tool set we have developed is similar in concept and spirit to the work of Nguyen but works with OPS5, the implementation language for the Transformation system.

III. OUR APPROACH TO THE VERIFICATION AND VALIDATION PROBLEM

The tool set we have developed consists of three major components each written entirely in Common Lisp: a parser, a matcher, and a rule analysis tool set. The first of the two components are specific to OPS5, while the tool set is not OPS5 specific.

The parser is a relatively straight forward recursive descent parser with reader macros introduced to make the lexical analysis easier, smoothing some of the syntactic irregularities of OPS5 (e.g. braces are changed to parentheses and so on) The output of the parser is a set of data structures that is given to the matcher in the form of a table.

The matcher is looking for patterns that are consistent and may match as the inferencing process proceeds. Obviously, the matcher can not answer the question of which rules actually fire as this is dependent on the data. Instead, the matcher looks at the table and sees which rules are coupled and identifies which rules may possibly interact. Generally two rules (say rule A and rule B) are coupled if a working memory element created, removed or altered by the actions in the RHS of rule A matches some condition element in the LHS side of rule B.

So for example, if there is a make action in the RHS of a rule, the matcher will attempt to see if there are any rules that have a condition element that may match the form of the working memory element that will be produced if the rule is triggered and fired. For the left hand side form to match the right hand side form for any rule clearly they must refer to the same element class. Since this is just given as a symbolic constant, this part of the matching process is trivial.

If the forms being compared in the matcher have the same class, the remaining parts of both the right hand side and left hand side form are divided into attribute value pairs and put into a canonical form. The RHS attributes not explicitly referenced in the **make** action are assigned the value **nil** in the canonical form. The values corresponding to attributes not explicitly referenced in the LHS form are given a value corresponding to a variable quantity (actually called a dummy variable). The attribute value pairs are sorted for both sides so that the matcher may continue simply by pairing off the values from the attribute value pairs and checking if the values match. As soon as one of the value pairs does not match, the matcher can conclude that the forms do not match because all attribute value pairs must be consistent.

The number of ways that two values for the same attribute may match in OPS5 is relatively limited. If both values are constant and they are **equal** then the values match. If either value is a variable then the values may match. If either of the values is **nil** then the values will match. If one of the values is a constant and the other is an OPS5 disjunction, one need only check that the constant is a member of the set of values that makes up the constant set for the disjunction. This is true because only constant values are allowed in OPS5 disjunctions. If both correspond to disjunctions then one need only consider if the sets of constants intersect. If either of the values correspond to a function call, then that value is treated as a variable. If the values being compared for consistency involve conjunctions then the conjunctions are compared on a clause by clause basis.

In OPS5 a clause is an ordered set consisting of a predicate and a value. (Note that constant c can be represented as the clause $\{= c\}$). Two clauses match if there is some value for which both clauses may be true. Thus if either of the value parts of the two clauses contain a variable there is some value to satisfy the clauses and thus they match. If both are constants then the relationship of the constants is checked to see if it is consistent with the predicates. For example, if one clause is $\{< 7\}$ and the other is $\{> 5\}$ then the clauses are consistent whereas given the clauses $\{< 1\}$ and $\{> 4\}$, the clauses are inconsistent and would not match. Now if we let the tuple

(relationship first-predicate second-predicate)

represent the two clauses. We may store the set of tuples corresponding to inconsistent clauses in a table with the tuple form used as a key. Since there are far fewer ways to get an inconsistency, the values are stored in a table called the ***inconsistency-table*** and if the tuple value is present when the table is searched, the clauses are deemed inconsistent. The set of all possible inconsistent tuples are as follows:

(= = <)	(< < >)	(< = =)	(< <= =)	(< <= >)	(< = >=)
(< = >)	(> > =)	(> = =)	(> > <=)	(= <= >)	(= = < >)
(= < =)	(= < >)	(> = <)	(> >= <)	(> = <=)	(> >= =)
(= > =)	(= > <)	(< < =)	(< < >=)	(= < >=)	(= >= <)
(> > <)	(= = >)	(= > <=)	(> >= <=)	(< <= >=)	

So to continue the example from above , the clauses {< 1} and {> 4} are inconsistent because the relationship of 1 to 4 is < and so the tuple representing the two clauses is given by (< < >) which appears in the table of inconsistent tuples.

Below is an example of two rules which would match by a make action. The rule make-default-pmdb-exposure-entry *directly affects* the rule set-wfpc-exposure. The matcher will match the make form in make-default-pmdb-exposure-entry to the form that is bound to <exposure-entry> following the matching rules which were outlined above:

```
(p make-default-PMDB-exposure-entry

(goal
  ^has-name      assign-PMDB-attributes
  ^has-status    active)

(assignment-record
  ^has-Pepsi-exposure-number <exp-number>
  ^has-exposure-id          <exp-id>
  ^has-alignment-id         <alignment-id>
  ^has-obset-id             <obset-id>
  ^is-last-exposure-in-alignment <> NIL
  ^is-last-exposure-in-obset   <> NIL )

(exposure-specification
  ^has-exposure-number      <exp-number>
  ^uses-SI-configuration    <SI-configuration>
  ^uses-SI-operating-mode   <SI-mode>
  ^has-exposure-time        <exposure-time>)

- (PMDB-exposure-entry
  ^has-exposure-id          <exp-id>
  ^has-alignment-id         <alignment-id>
  ^has-obset-id             <obset-id> )
-->

(make PMDB-exposure-entry
  ^has-exposure-id          <exp-id>
  ^has-alignment-id         <alignment-id>
  .
  .
  .
  ^uses-SI                  <SI-configuration>
  ^SI-observation-mode      nil
  ^is-a-point-source
  ^Pepsi-exposure-number    <exp-number> ) )
```

```

(p set-wfpc-exposure

  (goal
    ^has-name          assign-PMDB-attributes
    ^has-status        active)

  {<exposure-entry>
    (PMDB-exposure-entry
      ^uses-SI          << wfc pc >> ) }

-->

(modify <exposure-entry>
  ^uses-SI              wfpc ))

```

The matcher proceeds in a similar fashion if the action is a remove except that the matcher will never compare the LHS and RHS of the same rule for a remove. The rationale here is that an element cannot be removed in OPS5 on the RHS without being matched on the LHS so although the rule is coupled by a strict application of the definition, this information is of no particular use to the maintainer. Also, the matching rules for a remove deviate slightly from a make in that the matcher will only attend to attributes found explicitly in both the LHS and RHS forms.

The following example shows two rules which are coupled by a remove action. The matcher will find the rule make-goal-merge-alignments to be coupled with the rule find-parallel-with-mergeable-exposures:

```

(p make-goal-merge-alignments

  { <goal>
    (goal
      ^has-name      merge-exposures
      ^has-status    satisfied) }

  - (goal
      ^has-name      merge-alignments)

-->

  (remove <goal>)

  (make goal
    ^has-name      merge-alignments
    ^has-status    active
    ^task-list     find-potential-alignment-merges
                  insure-less-than-1296-alignments-per-obset
                  assign-alignment-attributes
                  assign-obset-orders ))

(p find-parallel-with-mergeable-exposures

  (goal
    ^has-name      merge-exposures
    ^has-status    active
    ^task-list     find-potential-exposure-merges)

  (exposure-specification
    ^has-exposure-number      <primary-exposure>

```

```

)

(exposure-link
  ^is-linked-to          <primary-exposure>
  ^has-link-type         parallel-with
  ^has-exposure-number   <parallel-exposure>
)

(exposure-specification
  ^has-exposure-number   <parallel-exposure>
)

(mergeable-level
  ^symbol                parallel-with
  ^value                 <parallel-with-level>)
-->

(make mergeable-exposures
  ^first-exposure-number   <primary-exposure>
  ^second-proposal-id     <parallel-proposal-id>
  ^second-version         <parallel-version>
  ^second-exposure-number <parallel-exposure>
  ^is-unmergeable         false
  ^is-mergeable-level     <parallel-with-level>
  ^merge-type             parallel-with
  ^has-unique-label       (genatom) ) )

```

The matching process for a modify action also uses the form matching algorithm elaborated above however only attempts this match if the form of the working memory element that will be created as a result of the modify is consistent with a left hand hand. By consistent, we mean that first of all, the RHS element which was altered as a result of the modify action must be present in the LHS of some other rule. If this is true, the matcher then checks to see if at least one of the attribute value pairs which was modified is present in the LHS element. If one is not, then there is no match. If one is present, then the matcher proceeds to check for any inconsistencies with the RHS element and the LHS element. The consistency checker for the modify is the same as the consistency checker for the make action with one slight exception. If an attribute exists in one element, but not in the other, the elements are still considered to be consistent.

The following two rules illustrate a rule coupling based on a modify action. The rule set-fos-exposure *directly affects* the rule set-fos-data-volume:

```

(p set-FOS-exposure

  (goal
    ^has-name      assign-PMDB-attributes
    ^has-status    active)

  {<exposure-entry>
    (PMDB-exposure-entry
      ^uses-SI      << FOS/BL FOS/RD >> ) }

-->

(modify      <exposure-entry>
  ^uses-SI    FOS ))

(p set-fos-data-volume

  (goal
    ^has-name      assign-PMDB-attributes

```

```

    ^has-status active)

{<exposure-entry>
(PMDB-exposure-entry
  ^has-exposure-id      <exp-id>
  ^has-alignment-id     <alignment-id>
  ^has-obset-id         <obset-id>
  ^uses-SI              fos
  ^expected-data-volume 0) }

(assignment-record
  ^has-Pepsi-exposure-number <exp-number>
  ^has-exposure-id          <exp-id>
  ^has-alignment-id         <alignment-id>
  ^has-obset-id             <obset-id>)

- (exposure-optional-parameters
  ^has-exposure-number      <exp-number>
  ^optional-parameter-name  read
  ^optional-parameter-value no)

-->

(modify <exposure-entry>
  ^expected-data-volume 100) )

```

The output of the matcher is a network of rule connections. As previously stated, this network can be visualized as a directed graph whose nodes represent rules and edges represent rule connections. Relatively simple graph transversal algorithms can be applied to the graph to answer pertinent verification and validation questions:

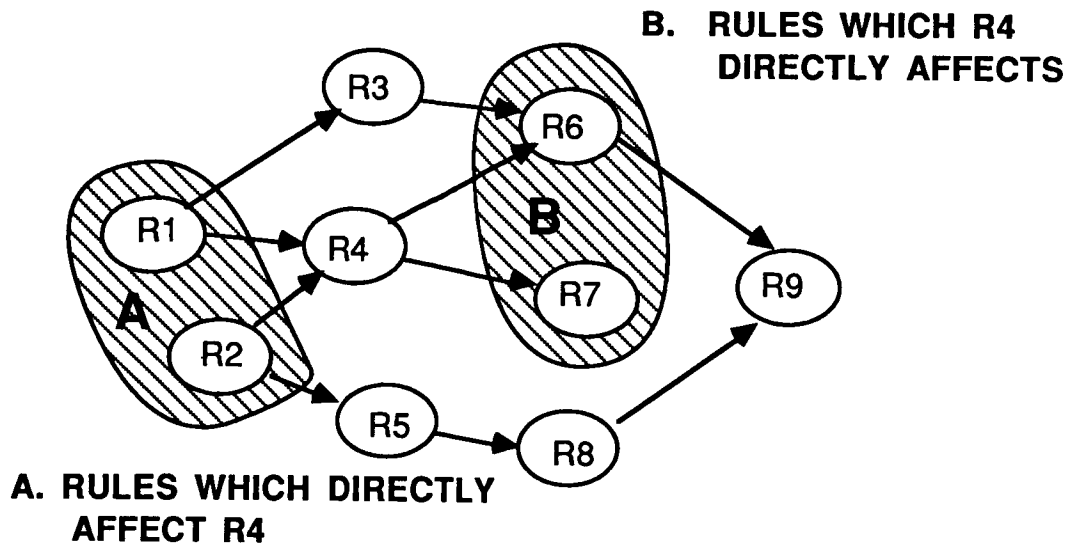
TOOLS FOR AIDING THE VALIDATION PROCESS

The following tools were developed to help in the validation process. These tools were created to reduce the number of rules and possible interactions which must be analyzed and tested when a modification is being made to the rulebase.

DIRECT EFFECTS

What rules are directly affected by rule A?
What rules directly affect A?

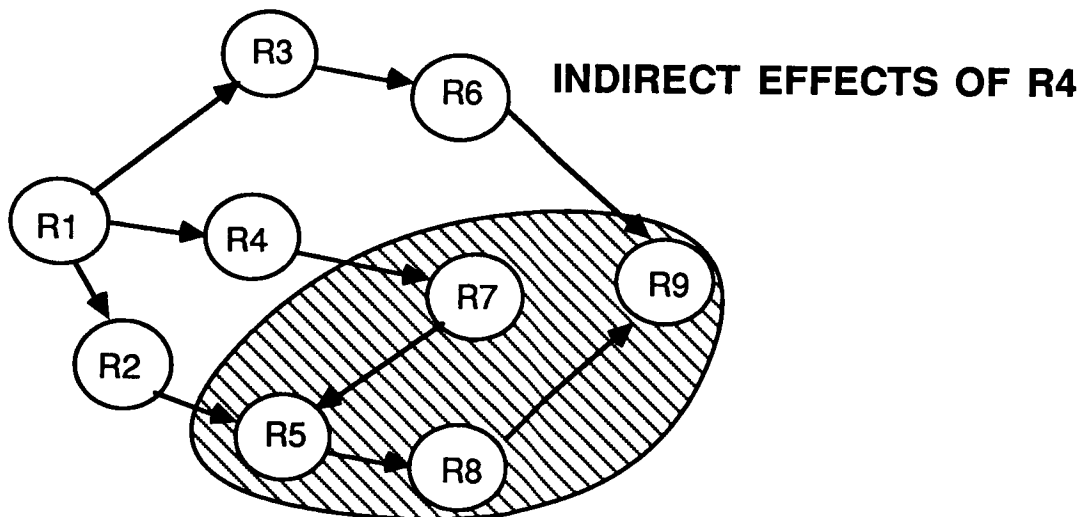
If we are able to partition the rulebase into smaller sections, we can minimize the knowledge needed to make the modification, and isolate the portion of the code which will require the most thorough testing. To find the direct affects of a rule we only have to perform a one level breadth first search on the graph of rule connections. It may also be of interest to know which rules directly affect a particular rule. The answer to this question can be answered by reversing the edges of the graph and performing the same one level breadth first search.



INDIRECT EFFECTS

Can the behavior of rule A ever affect rule B?

This tool is used for determining if the behavior of one rule can ever be affected by another rule. This is useful when investigating unpredictable outputs. For instance, you have just added Rule 1 to your rulebase, and you are getting unpredicted output which may be caused by Rule 9 firing and you want to know if adding rule 1 could be the cause of the problem. This question can be answered by a depth first search of the graph of rule connections.



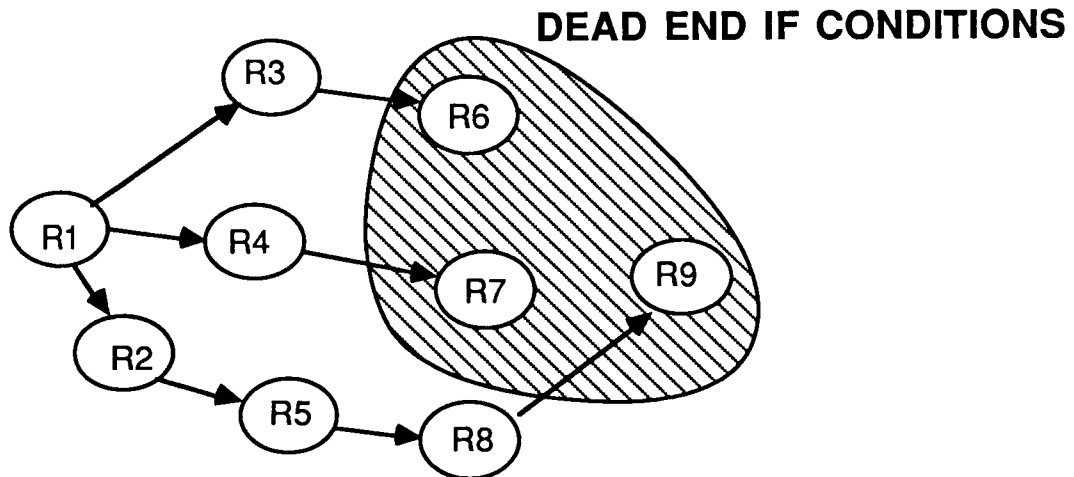
TOOLS FOR AIDING IN THE VERIFICATION PROCESS

The following tools were developed to find potential problems within the rulebase which may not be detected during source code testing.

DEAD END IF CONDITIONS

Which rules generate conclusions that are never used by any other rule in the rulebase?

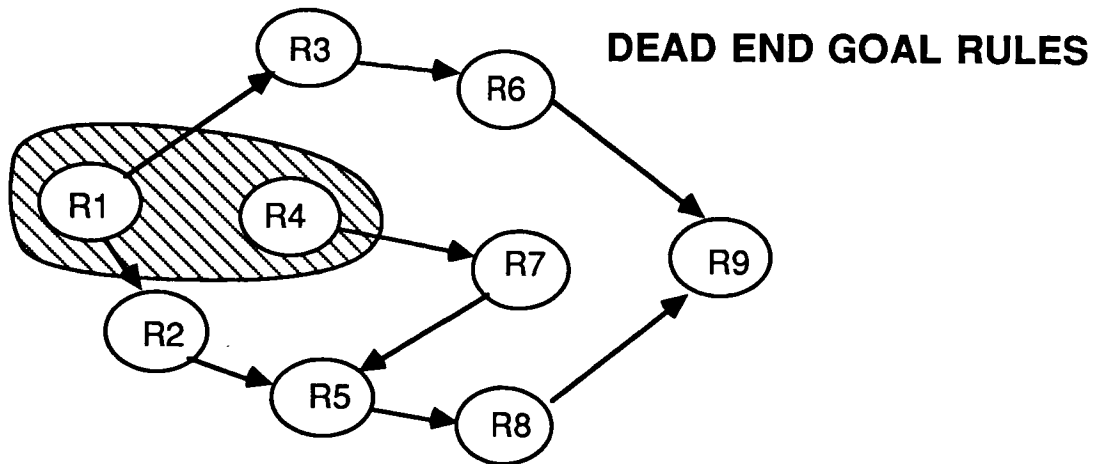
A dead end if condition is created when a rule generates a conclusion which is never used by another rule in the rulebase. This may mean that the rule needs to be removed because it is no longer necessary, or it may mean that an error was made when the rule was created. On the other hand, a dead end condition does not necessarily indicate a problem with the rulebase - it may be that these rules with dead end if conditions produce the final results of the problem. It is up to the maintainer to decide if the dead end if condition is a problem. Dead end if conditions can be identified easily by visiting each node in the graph and checking to see if the rule node has any adjacent rule nodes (a rule is adjacent to another rule if it can be reached in one step.) If it does not, that rule contains a dead end if condition.



DEAD END GOAL RULES

Which rules ask questions which are never answered by any rule in the rulebase?

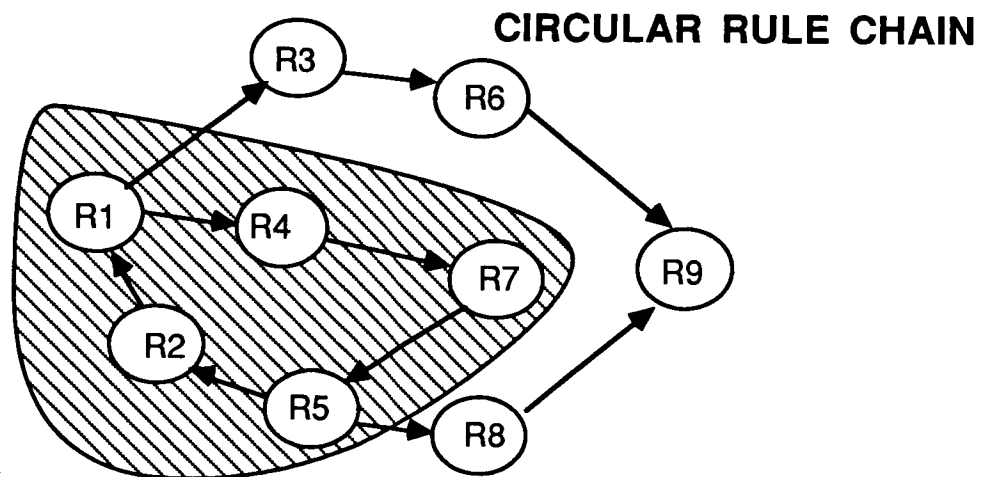
Dead end goal rules are created when the conditions of a particular rule are never satisfied by the actions of any rule in the rulebase. This may mean that the rule is outdated and needs to be removed or that an error was made when the rule was created. It may also be the case that the rule is directly fed by incoming data, and does not depend on the actions of some other rule. This tool identifies dead end goal rules, but it is up to the maintenance programmer to decide if they are a problem. To identify dead end goal conditions, the edges of the graph must be reversed, and each node in the graph must be visited. Like in the dead end if conditions, if the node is not adjacent to any other rule node, the rule contains a dead end goal.



CIRCULAR RULE CHAINS

Is there a possibility of an infinite loop occurring at run time? What rules might be involved?

This tool identifies the potential for entering in an infinite loop at run time. It may be that the rulebase system has a method for dealing with infinite loops, or it may be that there is a potential problem, and that no set of data previously identified this problem. This problem is identified by finding all strongly connected components within the rule connections graph. A strongly connected component is defined as a subgraph of a graph such that for every two nodes, x and y , there is a path from x to y and from y to x . [1]



IV. CONCLUSIONS

Soloway , Bachant, and Jensen [12] have noted that because of the dynamic nature of rules in OPS5, as the number of rules and people maintaining them grow, the chance that unwanted or unintended interactions between rules grows as well. The tool set we have developed provides a maintainer with a way to explicitly determine which rules may interact and in what ways. Thus the tool set provides a method for the maintainer to preserve a coherent rule base without having to be completely familiar with all rules or have access to someone that is such an expert in the knowledge base. Since the tool set is based on a type of dependency graph, it will work regardless of the underlying language that the graph represents. One need only write the parser and the matcher components which are language dependent. Furthermore, because the tool set is based on a rather simple graph data structure, it is readily extensible and when the need for another tool or type of analysis arises it may be easily integrated into the tool set.

REFERENCES

- [1] Baase, Sara, *Computer Algorithms - Introduction to Design and Analysis*, Addison-Wesley 1978, pp. 162.
- [2] Brownston, L., Farrell, R., Kant, E., and Martin, N., *Programming Expert Systems in OPS5*, Addison Wesley, 1985.
- [3] Chien, Y., Liebowitz, J., *Expert Systems in the SDI Environment*, Computer, Volume 19, No. 7, July 1986, pp.120 -121.
- [4] Cragun, B., Steudel, H., *A Decision-table-based Processor for Checking Completeness and Consistency in Rule-based Expert Systems*, International Journal Man-Machine Studies, Vol. 26, 1987, pp. 633-645.
- [5] Fairley, Richard E., *Software Engineering Concepts*, McGraw-Hill, 1985, pp.267-309.
- [6] Froscher, J., and Jacob, R., *Designing Expert Systems for Ease of Change*, IEEE Proceedings of the Expert systems in Government Symposium, October 1985, pp.246-251.
- [7] Hunter, Robin, *The Design and Construction of Compilers*, John Wiley and Sons, 1981, pp. 64-67.
- [8] Lindenmayer, K., Vick, S., and Rosenthal, D., *Maintaining an Expert System for the Hubble Space Telescope Ground Support*, Proceedings of 1987 Conference on Artificial Intelligence Applications, NASA Goddard Space Flight Center, May 13, 1987, pp. 1-12.
- [9] Nguyen, T., Perkins, W., Laffey, T., Pecora, D., *Knowledge Base Verification*, AI Magazine, Vol. 8, No. 2, Summer 1987, pp. 69-75.
- [10] O'Keefe, R., Balci, O., Smith, E., *Validating Expert System Performance*, IEEE Expert, Vol. 2, No. 1, Winter 1987, pp.81-90.
- [11] Rosenthal, D., Monger P., Miller, G., and Johnston, M., *An Expert System for Ground Support of the Hubble Space Telescope*, Proceedings of 1986 Conference on Artificial Intelligence Applications, NASA Goddard Space Flight Center, May 15, 1986, pp. 43-54.
- [12] Solloway, Elliot, Bachant, J., Jensen, K., *Assessing the Maintainability of XCON-in-RIME: Coping with the Problems of a VERY Large Rule-base*, Proceedings of 1987 AAAI Conference on Artificial Intelligence, Vol 2., July 13-17, 1987, pp. 824-829.
- [13] Winston, P., Horn, B., *LISP*, Addison-Wesley, 1984, pp. 169-175.

Shon Vick is a software developer in the Operations Software Branch at the Space Telescope Science Institute. He received a B.A. Degree in Economics/Mathematics from Rutgers College in 1980 and a M.Sc. Degree in Computer Science from the Whiting School of Engineering at The Johns Hopkins University.

Kelly Lindenmayer is a member of the technical staff of Computer Sciences Corporation in the Operations Software Branch at the STScI. She received an A.B. in Economics and Mathematics from Smith College in 1984 and is currently in pursuit of an M.Sc. in Computer Science from The Johns Hopkins University.

The Need For A Comprehensive Expert System
Development Methodology

by

John Baumert, Anna Critchfield, and Karen Leavitt
Computer Sciences Corporation
System Sciences Division
4600 Powder Mill Road
Beltsville, Maryland 20705

Abstract

In a traditional software development environment, the introduction of standardized approaches has led to higher quality, maintainable products on the technical side and greater visibility into the status of the effort on the management side. This study examined expert system development to determine whether it differed enough from traditional systems to warrant a reevaluation of current software development methodologies. Its purpose was to identify areas of similarity with traditional software development and areas requiring tailoring to the unique needs of expert systems. A second purpose was to determine whether existing expert system development methodologies meet the needs of expert system development, management, and maintenance personnel. The study consisted of a literature search and personal interviews. We determined that existing methodologies and approaches to developing expert systems are not comprehensive nor are they easily applied, especially to "cradle to grave" system development. As a result, we derived requirements for an expert system development methodology and arrived at an initial annotated outline for such a methodology. This work was conducted by Computer Sciences Corporation for the Goddard Space Flight Center under Contract NAS 5-27888, Task Assignment 357.

I. BACKGROUND

CSC and NASA have many years of experience working together on software development. Their successful, long-term relationship has led to a mutually agreed-to and understood approach to software development. This approach includes the management aspects of planning, monitoring, and controlling the development process as well as the technical aspects involved in actually developing software systems. From the experience gained in developing a prototype expert system Platform Management System (PMS) Resource Envelope Scheduling System (PRESS) and through conversations with other expert system developers--both government and contractors alike-- we identified a number of problems that pervade expert system development, including

- Difficulty in scheduling and planning to allow for radical changes to the system
- Difficulty in communicating between/among experts and knowledge engineers
- Difficulty in modularizing a production rule base
- Difficulty in quantifying and measuring progress and status
- Difficulty in devising a mechanism for validating and verifying the product
- Lack of guidance in the role of and mechanisms for quality assurance and configuration management of expert systems.

Soon after starting the assignment to develop PRESS, developers realized that the traditional management and technical approaches to software development did not meet the needs of expert system development. They also recognized that the documented expert system development approaches, while seemingly appropriate on paper, could not be readily applied in a real-world setting. This paper grew out of the realization that a comprehensive methodology that covers all aspects of expert system development is needed.

II. TERMINOLOGY

Terms in the field of expert system development are loosely and variously defined. To avoid misunderstanding, the following definitions are used throughout this paper:

Expert system--"A computer system designed to simulate the problem-solving behavior of a human who is expert in a narrow domain" (Denning, 1986)

Prototype--An early working model of a proposed system that does not exhibit all the system's features. For an expert system, a series of prototypes is used to

establish requirements (i.e., system behavior) as well as to demonstrate proof of concept.

Tool--Hardware and software packages used to assist the expert system development. These include languages, shells, knowledge acquisition programs, and hardware devices that host artificial- intelligence software packages.

Methodology--The framework that encompasses specific techniques and guidelines for defining and accomplishing work as well as guidance in planning and organizing to meet the requirements of the task. It also identifies the controls that must be implemented to monitor and assess the quality of the product and the progress toward completion.

III. APPROACH TO EXPERT SYSTEM DEVELOPMENT METHODOLOGY RESEARCH

The goal of this study was to identify and evaluate existing or proposed methods of developing expert systems. Our initial assumption was that this goal could be achieved primarily through a literature search and the subsequent evaluation and critique of documented methods. This assumption proved invalid. The literature search uncovered no comprehensive, formal methodology as defined above. This was especially true for a formalism describing expert system development from inception through operations and maintenance. Few expert systems, which represent a relatively new field in computer science, have completed their life cycle and, accordingly, documented experiences with the full life cycle are necessarily limited.

This finding indicated that the direction of the research needed to shift. What began as simple literature search evolved into a broader, three-tiered research project that painted a comprehensive picture of the state of expert system development and confirmed the need for a full-scale methodology. To determine whether an undocumented methodology exists, interviews and conferences were added to the research. The interviews reaffirmed the results of the literature search. The interviewees were able to discuss expert system development in general terms only, frequently admitting that their personnel followed no formal procedures and that management controls were noticeably lacking, or, as one interviewee phrased it, "management is a little loose." Conferences proved beneficial in that they exposed us to the most recent advances in the field.

IV. EXISTING APPROACHES TO EXPERT SYSTEM DEVELOPMENT

This section discusses documented approaches to expert system development. From the literature, the following papers were identified as being the most comprehensive--Buchanan et al. (1983), Waterman (1986), Bobrow, Mittal, and Stefik (1986), and Keller (1987). From the conferences, the seminar on software rapid prototyping (1987) and the tutorials by Zack (1987) and Martin (1987)

were the most comprehensive. A thorough discussion of results of each of the research approaches is found in Baumert, Critchfield, and Leavitt (1987).

Buchanan et al. (1983); Waterman (1986); and Bobrow, Mittal, and Stefik (1986) define essentially identical stages or phases in the expert system development (Table 1)--identification, conceptualization, formalization, implementation, and testing--from a technical perspective only. Management issues are not discussed.

Identification Phase--During this first phase, identification, the knowledge engineer and the expert identify the problem, the necessary resources, and the goals of the expert system.

Conceptualization Phase--During conceptualization, the knowledge engineer and the expert meet frequently to more accurately define the key concepts and relations as well as control mechanisms. The knowledge engineer records these concepts and relations to make the conceptual basis for problem formalization permanent.

Formalization Phase--Formalization involves expressing previously defined concepts and relations in an expert system building language. The knowledge engineer must understand the nature and structure of the knowledge to be captured by the system, and must select the tool(s) best suited to the application

Implementation Phase--During implementation, the formalized knowledge is turned into a working program, usually in a prototype environment in which programmers try various approaches until the prototype expert system appears to perform like the expert.

Testing Phase--The prototype is tested for both usefulness and performance. Each expert system requirement is verified and validated by a series of tests, thereby demonstrating that the knowledge representation is correct and that the inference engine reproduces the decision of the expert.

Knowledge Base Maintenance Phase--Maintenance activities are identical to those conducted in all the previous phases, but on a generally smaller scale. In addition, a plan that provides for system testing, development, transfer, and maintenance must be made.

Keller (1987) bases his expert system life cycle on the structured analysis techniques of Yourdon. Table 1 maps this life cycle into the life cycle discussed above. He associates nine activities with the life cycle--survey, structured analysis, knowledge base design, design, system integration, implementation, acceptance test, hardware analysis, and knowledge acquisition. Keller's list is deliberately not chronological; in fact, some activities must occur in parallel. For example, knowledge acquisition and structured analysis, which Keller views as nearly identical, are done at the same time and by the same people. He distinguishes the two by limiting knowledge acquisition to the functional, logical content of the expert's domain and structured analysis to the functional

components of peripheral activities such as the user interface. Structured analysis also includes the specification of the physical or technological components of the expert system.

Table 1 reveals an apparent degree of commonality among the various approaches; however, although the phases seem analogous, the activities within them are not necessarily the same. For example, the testing stage given by Buchanan et al. and Waterman refers to the testing of the original prototype whereas that of Keller is broader and contains integration tests.

Martin (1987) has developed the Expert System Controlled Iterative Enhancement (ESCIE) methodology. ESCIE has seven stages--initial feasibility study, rapid prototype demonstration, basic system usage, scope development system, refinement/enhancement, productization, and operations/maintenance. Within the middle three stages, Martin places five phases--requirements analysis, specification-model, architectural design, design-implement-test, and evaluation. These phases introduce iterativeness to ESCIE.

Martin describes each of her seven stages as follows. During the feasibility study stage, the feasibility and advisability of working on a specific problem is determined. The rapid prototype demonstration stage is used to illustrate the problem-solving capability of the expert system and to demonstrate the ability to obtain an executable system rapidly. Its purpose is to garner the support of management, the user, and the expert. It can be performed concurrent with the next stages to permit investigation of different knowledge base designs or difficult aspects of the problem solution. The basic system usage stage demonstrates that the expert system can perform the required reasoning and be beneficial to the users. The scope development system stage demonstrates the system's utility and performance over the scope of the desired functionality, while the refinement/enhancement stage is devoted to improving the system's performance, usability, capacity, and functionality. Ease of maintenance is emphasized during this stage. The productization stage is used to produce a marketable product and the operations/maintenance stage provides for operations support, corrective action, and enhancements in response to changing environments.

Martin describes each of her five phases as follows. Her requirements analysis phase is identical to traditional software requirements analysis during which the user's requirements are determined, acceptance criteria established, and the feasibility and advisability of proceeding with the project are evaluated. The specification-model phase defines the problem solving required of the expert system. The architectural design phase determines the major components of the expert system and their structure and interfaces. This phase applies not only to the structure of the knowledge base but also to the inference engine. The goal of the design-implement-test phase is to obtain a working version of the system as early as possible so the user can validate it as it grows. Design decisions are implemented and evaluated immediately. As design and coding progress, the system is verified through unit and integration tests. Ultimately,

Table 1. Comparison of Four Approaches to Expert System Development

BUCHANAN ET AL. WATERMAN	BOBROW, MITTAL, AND STEFIK	KELLER	MARTIN
IDENTIFICATION CONCEPTUALIZATION	IDENTIFICATION CONCEPTUALIZATION	SURVEY STRUCTURED ANALYSIS KNOWLEDGE ACQUISITION HARDWARE ANALYSIS	REQUIREMENTS ANALYSIS SPECIFICATION-MODEL
FORMALIZATION		DESIGN KNOWLEDGE ACQUISITION KNOWLEDGE BASE DESIGN	ARCHITECTURAL DESIGN
IMPLEMENTATION	PROTOTYPING CREATING USER INTERFACES	IMPLEMENTATION	DESIGN-IMPLEMENT-TEST
TESTING	TESTING AND REDEFINITION KNOWLEDGE BASE MAINTENANCE	ACCEPTANCE TEST SYSTEM INTEGRATION	EVALUATION

G632-3/12-87/61

the system is evaluated for its reasoning capabilities. smoothness of interfaces, visibility, ease of enhancement, performance, reliability, utility, cost effectiveness, and scope. These phases are mapped to those of Buchanan et al.; Waterman; Bobrow, Mittal, and Stefik; and Keller in Table 1

Unlike others, Martin's methodology integrates management and technical issues. ESCIE strives to provide the often lacking but necessary management control to prototyping and yet maintain technical creativity. She defines estimation and computation approaches, cost/schedule drivers, project roles, job descriptions, the types and descriptions of documents needed, and guidelines for time and manpower required for the rapid prototype through refinement stages for different sized expert systems.

V. COMPARISON OF WATERFALL AND EXPERT SYSTEM APPROACHES

For NASA, expert systems are likely to be subsets of larger systems developed according to the traditional waterfall model of software development. Therefore, an expert system development methodology may be required to adopt either directly or by reference aspects of this traditional methodology. Zack (1987) emphasized that at least 50 percent, if not more, of the time spent on expert system development involves the traditional activities of gathering data and coding.

Given the familiarity with and overall acceptance of the waterfall model, we concluded that some aspects of the waterfall model should be adopted. This conclusion was based on the fact that the life-cycle phases in the waterfall model are well defined and that most interpretations of it, recognizing the importance of an integrated management and technical approach to software development, emphasize defining activities within each life-cycle phase, planning and replanning these activities, and monitoring and controlling the whole process. Even a preliminary assessment of expert system methodology needs indicates that both management and technical issues must be addressed.

The waterfall model used in this study, and shown in Figure 1, represents that used by the Mission Operations and Data Systems Directorate at NASA/Goddard Space Flight Center. It is divided into six distinct phases: requirements analysis, design, implementation, integration and test, acceptance test, and maintenance.

Our analysis revealed that many activities in the traditional waterfall method are analogous to those outlined for expert system development and defined in Table 1. We reexamined the expert system development approaches in the context of the waterfall method, identifying both similarities and differences. Table 2 summarizes the results of this analysis, listing only the major activities within the traditional software development methodology and relating them, where possible, to analogous or parallel expert system development activities. For example, knowledge acquisition and knowledge representation were found

Table 2. Comparison of Traditional and Expert System Software Life Cycles

TRADITIONAL SOFTWARE SYSTEM	EXPERT SYSTEM
<p>REQUIREMENTS ANALYSIS</p> <ul style="list-style-type: none"> REVIEW REQUIREMENTS FOR COMPLETENESS DEFINE ADDITIONAL REQUIREMENTS REMOVE REDUNDANT/INACCURATE REQUIREMENTS SOFTWARE REQUIREMENTS REVIEW <p>DESIGN</p> <ul style="list-style-type: none"> TOP-DOWN STRUCTURED DESIGN <p>DESIGN REVIEWS</p> <p>IMPLEMENTATION</p> <ul style="list-style-type: none"> CODE SYSTEM USING PROGRAMMING LANGUAGE UNIT TEST BUILD APPROACH <ul style="list-style-type: none"> INCREMENTAL IMPLEMENTATION OF REQUIREMENTS SUCCESS = ACCEPTANCE OF THE SYSTEM <p>VERIFICATION AND VALIDATION</p> <ul style="list-style-type: none"> UNIT TESTS MODULE TESTS INTEGRATION TESTS INTERFACE TESTS ACCEPTANCE TESTS <p>OPERATIONS AND MAINTENANCE</p> <ul style="list-style-type: none"> CORRECT ERRORS ENHANCE SYSTEM CONFIGURATION MANAGEMENT <ul style="list-style-type: none"> SOFTWARE DOCUMENTATION 	<p>KNOWLEDGE ACQUISITION</p> <ul style="list-style-type: none"> GATHER KNOWLEDGE REFINE KNOWLEDGE COMMUNICATION BETWEEN EXPERT AND KNOWLEDGE ENGINEER <p>KNOWLEDGE REPRESENTATION</p> <ul style="list-style-type: none"> KNOWLEDGE BASE <ul style="list-style-type: none"> CONTAINS FACTS AND RULES KNOWLEDGE REPRESENTED THROUGH RULES, FRAMES, SEMANTIC NETS COMMUNICATION BETWEEN EXPERTS, KNOWLEDGE ENGINEER, PROGRAMMERS <p>IMPLEMENTATION</p> <ul style="list-style-type: none"> "CODE" SYSTEM USING AI TOOLS "UNIT" TEST ITERATE OVER KNOWLEDGE ACQUISITION, AND REPRESENTATION, REQUIREMENTS ANALYSIS, DESIGN, AND CODING SUCCESS = KNOWLEDGE GAINED REGARDING KNOWLEDGE REPRESENTATION AND DESIGN <p>VERIFICATION AND VALIDATION</p> <ul style="list-style-type: none"> UNIT TESTS MODULE TESTS INTEGRATION TESTS INTERFACE TESTS ACCEPTANCE TESTS <p>OPERATIONS AND MAINTENANCE</p> <ul style="list-style-type: none"> CORRECT ERRORS ENHANCE SYSTEM, INCLUDING CHANGING KNOWLEDGE CONFIGURATION MANAGEMENT <ul style="list-style-type: none"> SOFTWARE DOCUMENTATION KNOWLEDGE BASE

G632-4/12-87[6]

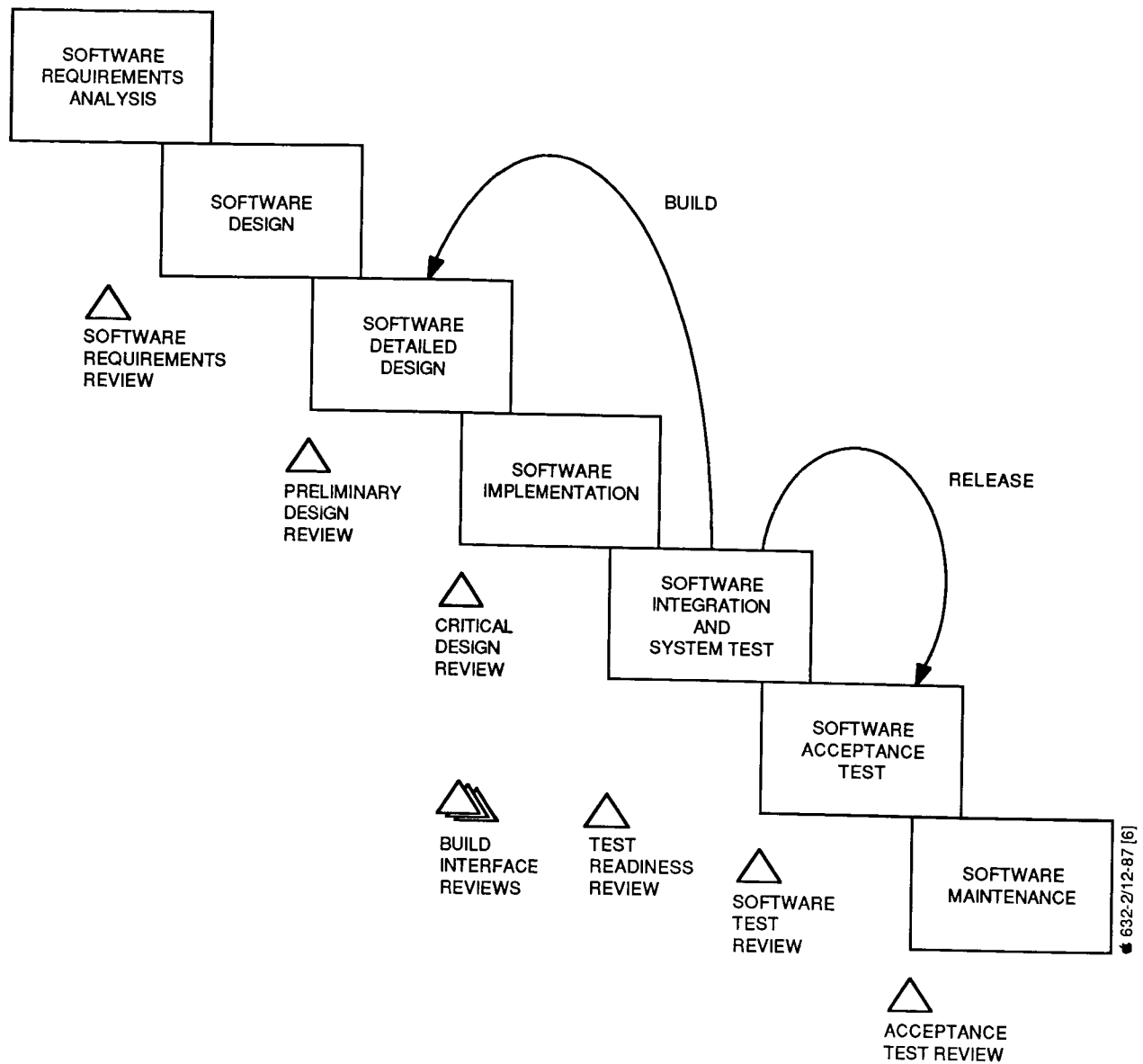


Figure 1. The Software Life Cycle--Waterfall Method

to be analogous to, but also much broader than, requirements analysis and design. Another difference lies in the timing of activities. Traditional software development is a sequential process, whereas expert system development is not chronologically ordered i.e., knowledge acquisition, knowledge representation, and implementation can and usually do occur in parallel.

VI. REQUIREMENTS OF AN EXPERT SYSTEM DEVELOPMENT METHODOLOGY

As a result of our research, we arrived at a number of requirements that any methodology for developing an expert system must satisfy. In addition to the literature search results, information gained from interview and conferences has been used to arrive at these requirements. These requirements do not comprise a distinct methodology, rather they represent key elements that such a methodology must contain. Several different methodologies could be derived from them. We consider them to form a bridge between the research effort discussed in this paper and an outline for an expert system development methodology.

As shown in Table 2, similar activities take place in expert system development and traditional software development. Therefore, we made two major assumptions. First, the waterfall method is a viable first step in developing an expert system methodology and, second, expert system development comprises a mix of expert-system-unique and traditional software development activities. Since the waterfall method has proven reasonably successful, we concluded that it should not be abandoned totally for expert system development. The basic concepts of the waterfall method do require modification to accommodate such expert system development activities as prototyping.

These requirements are not given in any specific order; the emphasis is on concepts rather than timing or importance. An initial attempt to divide this list along management and technical lines also proved impractical since the requirements too often crossed artificial boundaries.

Iteration is a key element throughout the development process, especially during implementation and testing. For example, testing may reveal that the system needs to be refined, redesigned, or reformulated. Reformulation entails changes in the identification and/or conceptualization phase that, in turn, affect the remaining phases. Redesign occurs in the formalization phase by changing the representation of the knowledge. Failure to meet performance requirements may also force a redesign. Refinement occurs via iterations throughout implementation and testing when relatively minor changes occur.

An expert system development methodology must satisfy the following requirements.

1. Include a special section on the definition of terms.

2. Establish guidelines for evaluating a problem for its applicability to an expert system solution.
3. Define the role of each member of the development team.
4. Make the expert an active (and willing) member of the development team.
5. Involve users in evaluating the expert system. The time spent in this activity will not only result in a system more satisfactory to the user, it will also lessen the training time needed upon delivery.
6. Require the assignment of a knowledge engineer whose goal is to build a system that satisfies both the expert and the user.
7. Through prototyping, allow requirements definition and expert system software development to proceed in parallel.
8. Provide for frequent prototype demonstrations. This fosters a close working relationship between the development team (including the expert) and users, uniting them in a common goal.
9. Provide for control that allows rapid changes but restricts "free lancing." Establish a way of tracing a system's evolution.
10. Contain recommendations on how to schedule expert system development and determine when to finalize the requirements, i.e., terminate the prototyping process. Management--both customer and developer--must mutually agree to an initial estimate of this date at the beginning of the project.
11. Provide an objective means of reporting progress. Terms such as progress, productivity, rate of progress, and completion of a schedule milestone may have to be redefined to allow for the iterative nature of expert system development.
12. Establish a set of baseline documents to be created initially in the early phases of expert system development, updated with each prototype demonstration, and formally delivered at the end of the system development.
13. Provide guidelines for scheduling and budgeting that allow for contingency planning.
14. Establish a set of standards and procedures encompassing knowledge acquisition and programming techniques during the actual creation of an expert system, stressing modularity and a structured approach.
15. Allocate an active role for quality assurance during expert system development.
16. Establish open lines of communication throughout the project.

VII. FUTURE WORK

The next logical step is to derive a methodology that satisfies the requirements listed above. We recommend using, and modifying where necessary, the traditional software development approach as the starting point for the new expert system development methodology. After the expert system development methodology is reasonably well established, identify suitable project(s) to implement all or parts of the methodology. The more controversial aspects of the methodology may be implemented separately for refinement before the entire methodology is used on a single project.

REFERENCES

Waterman, D. A., 1986, A Guide to Expert Systems, Reading, MA: Addison-Wesley.

Buchanan, B., 1983, Barstow, D., Bechtel, R., Bennett, J., Coancey, W., Kulikowski, C., Mitchell, T., and Waterman, D. A., "Constructing an Expert System," Building Expert Systems, F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, eds., Reading, MA: Addison-Wesley, pp. 127-167.

Keller, R. 1987, Expert System Technology: Development and Application, Englewood Cliffs, NJ: Yourdon Press.

Denning, P. J., 1986, "Towards a Science of Expert Systems," IEEE Expert, vol. 1, no. 2, pp. 80-83.

Bobrow, D. G., 1986, S. Mittal, M. J. Stefik, "Expert Systems: Perils and Promise," ECommunications of the ACMR, vol. 29, no. 9, pp. 880-894.

Connell, J., 1987, "Software Rapid Prototyping seminar, August 27-28, Washington, DC.

Zack, B. A., 1987, "Building Operational Expert Systems," tutorial presented at Third Annual Expert Systems in Government Conference, October, Washington, DC.

Martin, N., 1987, "The Management of Expert System Development," tutorial presented at Third Annual Expert Systems in Government Conference, October, Washington, DC.

Baumert, J., 1987, A. Critchfield, K. Leavitt, Expert System Development Methodology Study Report, CSC/TM-87/6728.

BIBLIOGRAPHIC DATA SHEET

1. Report No. NASA CP-3009	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle 1988 Goddard Conference on Space Applications of Artificial Intelligence		5. Report Date August 1988	
		6. Performing Organization Code 500	
7. Author(s) James Rash and Peter Hughes, Editors		8. Performing Organization Report No. 88B0212	
9. Performing Organization Name and Address Mission Operations & Data Systems Directorate NASA/GSFC Greenbelt, MD 20771		10. Work Unit No.	
		11. Contract or Grant No.	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546		13. Type of Report and Period Covered Conference Publication	
		14. Sponsoring Agency Code 500	
15. Supplementary Notes			
16. Abstract <p>This publication comprises the papers presented at the 1988 Goddard Conference on Space Applications of Artificial Intelligence held at the NASA/Goddard Space Flight Center, Greenbelt, Maryland on May 24, 1988. The purpose of this annual conference is to provide a forum in which current research and development directed at space applications of artificial intelligence can be presented and discussed. The papers in this proceedings fall into the following areas: Mission Operations Support, Planning & Scheduling, Fault Isolation/Diagnosis, Image Processing & Machine Vision, Data Management, Modeling & Simulation, and Development Tools/Methodologies.</p>			
17. Key Words (Selected by Author(s)) Artificial Intelligence, expert systems, mission operations support, planning and scheduling, fault isolation, fault diagnosis, image processing, machine vision, data management, modeling, simulation.		18. Distribution Statement Unclassified - unlimited Subject Category - 63	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 464	22. Price* A20

*For sale by the National Technical Information Service, Springfield, Virginia

22161

GSFC 25-44 (10/77)

NASA-Langley, 1988